

**Máster Universitario en Computación Paralela y Distribuida
Algoritmos Paralelos Matriciales en Ingeniería**



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

Librerías Matriciales de Altas Prestaciones

Contenido

- Introducción: De los algoritmos a las librerías
- Librerías matriciales y arquitecturas
- Librerías de Álgebra Lineal Numérica
 - LAPACK
 - ScaLAPACK
 - La colección ACTS
 - StructPack
- Implementación de la Descomposición QR en LAPACK y ScaLAPACK: aproximación secuencial y paralela
- Caso de estudio: Detectando puntos luminosos en presencia de la Radiación Cósmica de Fondo del Universo
- Conclusiones

- *De los algoritmos a las librerías: Un ejemplo sencillo*

$$\|x\| = \sqrt{x_1^2 + x_2^2 + \dots + x_n^2}, \quad x \in \Re^n$$

Algoritmo:

```

Norm =  $x_1^2$ 
for  $i = 2, 3, \dots, n$ 
    Norm = Norm +  $x_i^2$ 
end
Norm = sqrt(Norm)

```

El algoritmo fracasa si se quiere calcular, en simple precisión, la norma del vector

$$x = [10^{20} \ 10^{21} \ 10^{22}]^T, \quad x \in \Re^3$$

Solución

$$\|x\| = x_{\max} \sqrt{\left(\frac{x_1}{x_{\max}}\right)^2 + \left(\frac{x_2}{x_{\max}}\right)^2 + \dots + 1 + \dots + \left(\frac{x_n}{x_{\max}}\right)^2}, \quad x \in \Re^n$$

$$x_{\max} = \max_i |x_i|$$

• Implementación del BLAS 1

```
REAL FUNCTION SNRM2(N,X,INCX)
* .. Scalar Arguments ..
INTEGER INCX,N
* ..
* .. Array Arguments ..
REAL X(*)
* ..
*
* Purpose
* ======
*
* SNRM2 returns the euclidean norm of a vector via the function
* name, so that
*
* SNRM2 := sqrt( x*x ).
*
* Further Details
* =====
*
* -- This version written on 25-October-1982.
* Modified on 14-October-1993 to inline the call to SLASSQ.
* Sven Hammarling, Nag Ltd.
*
* =====
*
* .. Parameters ..
REAL ONE,ZERO
PARAMETER (ONE=1.0E+0,ZERO=0.0E+0)
* ..
* .. Local Scalars ..
REAL ABSXI,NORM,SCALE,SSQ
INTEGER IX
* ..
* .. Intrinsic Functions ..
INTRINSIC ABS,SQRT
```

• Implementación del BLAS 1

```

* ..
IF (N.LT.1 .OR. INCX.LT.1) THEN
    NORM = ZERO
ELSE IF (N.EQ.1) THEN
    NORM = ABS(X(1))
ELSE
    SCALE = ZERO
    SSQ = ONE

DO 10 IX = 1,1 + (N-1)*INCX,INCX
    IF (X(IX).NE.ZERO) THEN
        ABSXI = ABS(X(IX))
        IF (SCALE.LT.ABSXI) THEN
            SSQ = ONE + SSQ* (SCALE/ABSXI)**2
            SCALE = ABSXI
        ELSE
            SSQ = SSQ + (ABSXI/SCALE)**2
        END IF
    END IF
10 CONTINUE
    NORM = SCALE*SQRT(SSQ)
END IF
*
SNRM2 = NORM
RETURN
*
* End of SNRM2.
*
END

```

$$\begin{aligned}
& 1 + \left[1 + \left(\frac{x_2}{x_1} \right)^2 + \left(\frac{x_3}{x_1} \right)^2 \right] \left(\frac{x_1}{x_4} \right)^2 \\
&= \left[\left(\frac{x_1}{x_4} \right)^2 + \left(\frac{x_2}{x_4} \right)^2 + \left(\frac{x_3}{x_4} \right)^2 + 1 \right]
\end{aligned}$$

Beneficios

- Robustez
- Precisión
- Eficiencia

¿Por qué usar librerías matriciales?

- Desde los principios de la computación científica se entendió la necesidad de utilizar librerías numéricas y matriciales
- Objetivos:
 - Legibilidad
 - Eficiencia
 - Portabilidad
 - Fiabilidad: calidad de código
 - No reprogramar códigos sencillos: organización por niveles

Librerías matriciales y arquitecturas

•**LAPACK: Librería secuencial/paralela (memoria compartida).**
(1992)

Conjunto de subrutinas que resuelven:

Sistemas lineales, Mínimos cuadrados, Valores y vectores propios y singulares, Descomposiciones matriciales

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

Implementation Guide for LAPACK

UT, CS-90-101, April 1990.

E. Anderson and J. Dongarra

LAPACK: A Portable Linear Algebra Library for High-Performance Computers

UT, CS-90-105, May 1990.

E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. DuCroz, A. Greenbaum, S. Hammarling, A. McKenney, D. Sorensen

Librerías matriciales y arquitecturas

LAPACK: Librería secuencial/paralela (memoria compartida). (1992)

Filosofía: Algoritmos orientados a bloques

- Basados en BLAS
- Eficiencia
- Portabilidad

Tipos de matrices:

- Densas.
- Banda.
- Reales y complejas.

Utilizable sobre:

- Computadores secuenciales.
- Multiprocesadores con Memoria compartida

BLAS

Basic Linear Algebra Subprograms

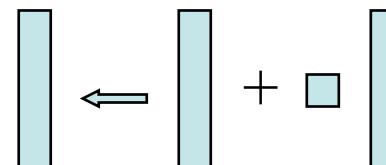
Librería de rutinas que implementan versiones de las rutinas elementales denominadas núcleos computacionales por su amplia utilización .

Objetivos:

- Portabilidad
- Altas prestaciones y escalabilidad
- Facilidad de uso

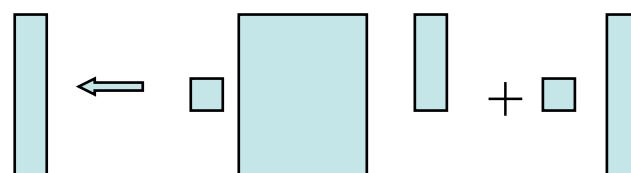
- Nivel 1: operaciones Vector-Vector

$$y \leftarrow \alpha x + y \quad O(n)$$



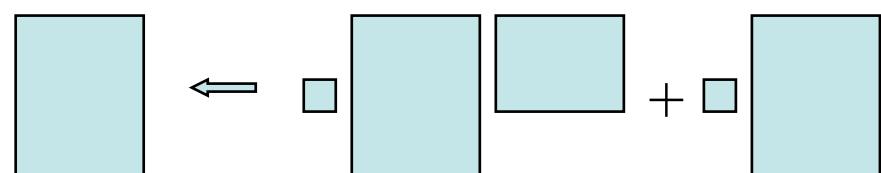
- Nivel 2: operaciones Matriz-Vector

$$y \leftarrow \alpha A x + \beta y \quad O(n^2)$$



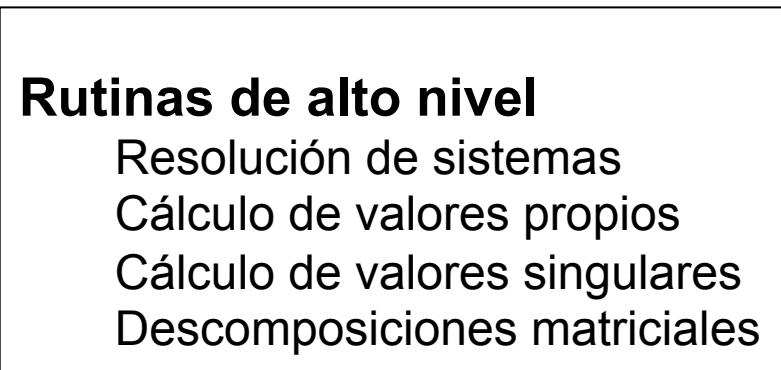
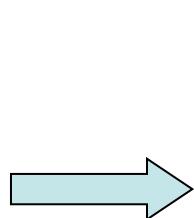
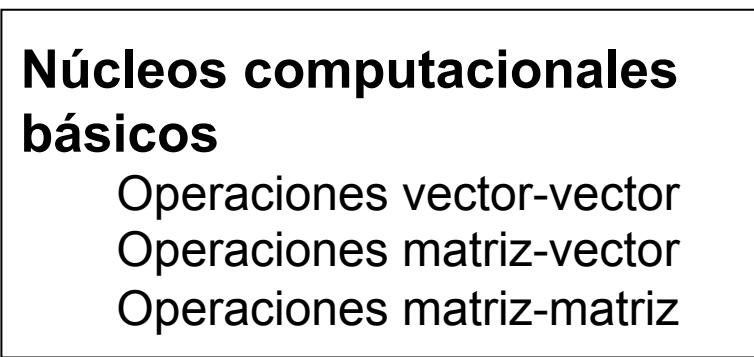
- Nivel 3: operaciones Matriz-Matriz

$$C \leftarrow \alpha A B + \beta C \quad O(n^3)$$



LAPACK: Librería secuencial/paralela (memoria compartida). (1992)

Se trata de emplear como filosofía de diseño:



BLAS → **LAPACK**

**Basic Linear
Algebra
Subprograms**

Linear Algebra PACK

Para diseñar algoritmos tipo LAPACK hay que organizar el problemas de forma tal que pueda ser resuelto mediante llamadas a los núcleos computacionales básicos.

LAPACK: Librería secuencial/paralela (memoria compartida). (1992)

Tipos de rutinas:

Rutinas Driver: Resuelve un problema completo (sistemas de ecuaciones)

Rutinas computacionales: Realizan una tarea computacional (Desc. LU)

Rutinas auxiliares: Realizan una subtarea o trabajo de menor nivel.

Formato de rutinas driver y computacionales: XYYZZZ

X: Tipo de datos:

S : REAL

D : DOUBLE PRECISION

C : COMPLEX

Z : DOUBLE COMPLEX

YY: Tipo de matriz

BD bidiagonal

GB general band

GE general (no simétrica, rectangular,...)

.....

ZZZ: Operación:

SV: sistemas de ecuaciones

EV: valores propios ...

LAPACK: Librería secuencial/paralela (memoria compartida). (1992)

Información sobre las rutinas de LAPACK:

LAPACK WORKING NOTES → LAWN

<http://www.netlib.org/lapack/lawns/downloads/>

LAPACK: Librería secuencial/paralela (memoria compartida). (1992)

Actualmente existen versiones del BLAS que se pueden ejecutar utilizando threads. Esto permite utilizar el LAPACK (construido sobre BLAS) de forma sencilla y eficiente sobre computadores de tipo multicore.

Ejemplo: La librería **mkl** de intel (<http://software.intel.com/en-us/intel-mkl>) tiene incorporada esta funcionalidad.

También existen versiones de BLAS y LAPACK para GPU (Graphics Processing Units). Estos aceleradores disponen de un número elevado de cores y permiten un paralelismo masivo.

Ejemplos:

cuBLAS: Implementación de los núcleos computacionales BLAS escritos en CUDA (<https://developer.nvidia.com/cublas>).

MAGMA: Implementación de LAPACK para GPU (<https://developer.nvidia.com/magma>).

MAGMA: **Matrix Algebra on GPU and Multicore Architectures**

The interface for MAGMA is similar to LAPACK.

Many routines have the same base names and the same arguments as LAPACK.

There are several classes of routines in MAGMA:

Driver routines – Solve an entire problem.

Computational routines – Solve one piece of a problem.

BLAS routines – Basic Linear Algebra Subroutines. These form the basis for linear algebra algorithms.

Auxiliary routines – Additional BLAS-like routines, many originally defined in LAPACK.

Utility routines – Additional routines, many specific to GPU programming.

MAGMA: Matrix Algebra on GPU and Multicore Architectures

In general, MAGMA follows LAPACK's naming conventions. The base name of each routine has a one letter precision (occasionally two letters), two letter matrix type, and usually a 2-3 letter routine name. For example, DGETRF is D (double-precision), GE (general matrix), TRF (triangular factorization).

A brief summary of routines is given here.

Suffix	Example	Description
none	magma_dgetrf	hybrid CPU/GPU routine where the matrix is initially in CPU host memory.
_m	magma_dgetrf_m	hybrid CPU/multiple-GPU routine where the matrix is initially in CPU host memory.
_gpu	magma_dgetrf_gpu	hybrid CPU/GPU routine where the matrix is initially in GPU device memory.
_mgpu	magma_dgetrf_mgpu	hybrid CPU/multiple-GPU routine where the matrix is distributed across multiple GPUs' device memories.

MAGMA:

Matrix Algebra on GPU and Multicore Architectures

Driver routines solve an entire problem.

Name	Description
gesv, posv	solve linear system, $AX = B$
gels	least squares solve, $AX = B$
geev, syev, heev	eigenvalue solver, $AX = X \Lambda$
syevd, heevd	eigenvalue solver using divide & conquer
sygvd, hegvd	generalized eigenvalue solver, $AX = BX \Lambda$
gesvd	singular value decomposition (SVD), $A = U \Sigma V^H$
gesdd	SVD using divide & conquer

MAGMA:

Matrix Algebra on GPU and Multicore Architectures

Computational routines

Name	Description
: Triangular factorizations :	Description
getrf, potrf	triangular factorization (LU, Cholesky)
getrs, potrs	triangular forward and back solve
getri, potri	triangular inverse
getf2, potf2	triangular panel factorization (BLAS-2)
. Orthogonal factorizations	Description
ge{qrf, qlf, lqf, rjf*}	QR, QL, LQ, RQ factorization
geqp3	QR with column pivoting (BLAS-3)
or{mqr, mql, mlq, mrq*}	multiply by Q after factorization (real)
un{mqr, mql, mlq, mrq*}	multiply by Q after factorization (complex)
or{gqr, gql*, glq*, grq*}	generate Q after factorization (real)
un{gqr, gql*, glq*, grq*}	generate Q after factorization (complex)
geqr2	QR panel factorization (BLAS-2)
. Eigenvalue & SVD	Description
gehrd	Hessenberg reduction (in geev)
sytrd, hetrd	tridiagonal reduction (in syev, heev)
gebrd	bidiagonal reduction (in gesvd)

MAGMA: **Matrix Algebra on GPU and Multicore Architectures**

BLAS routines

- BLAS routines follow a similar naming scheme: precision, matrix type (for level 2 & 3), routine name.
- For BLAS routines, the **magma_ prefix** indicates a wrapper around CUBLAS (e.g., magma_zgemm calls cublasZgemm),
- while the **magmablas_ prefix** indicates the own MAGMA implementation (e.g., magmablas_zgemm).
- All MAGMA BLAS routines are GPU native and take the matrix in GPU memory.

cuBLAS (<http://docs.nvidia.com/cuda/cublas/#axzz3LK1SnZST>)

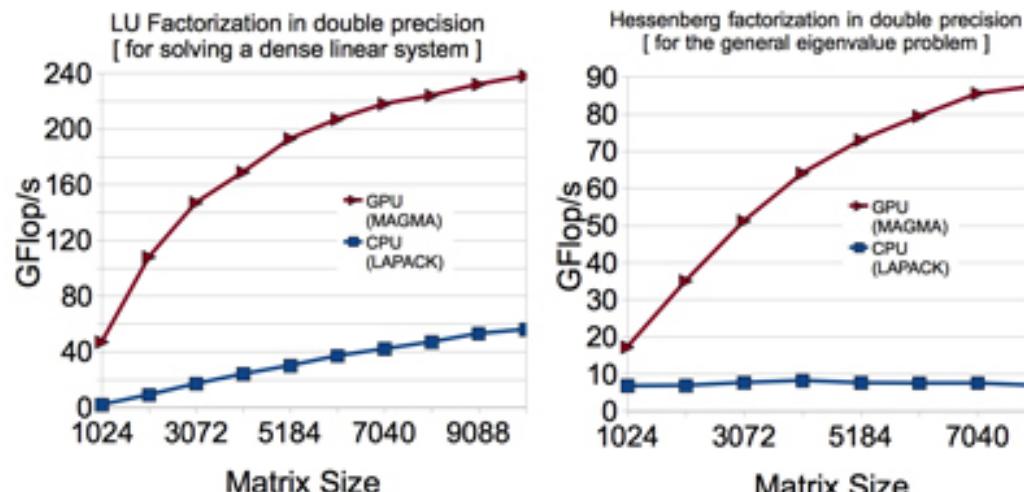
- The NVIDIA CUDA Basic Linear Algebra Subroutines (cuBLAS) library is a GPU-accelerated version of the complete standard BLAS library that delivers 6x to 17x faster performance than the latest MKL BLAS. New in CUDA 6.0 is multi-GPU support in cuBLAS-XT.
- Heterogeneous LAPACK implementations such as CULA Tools and MAGMA use the GPU-accelerated BLAS routines in the cuBLAS library, .

MAGMA: Matrix Algebra on GPU and Multicore Architectures

Key Features

- Excellent performance and high accuracy (LAPACK compliant)
- Multiple precision arithmetic support (S/D/C/Z)
- Hybrid algorithms using both multicore CPUs and GPUs

Accelerating Dense Linear Algebra with GPUs

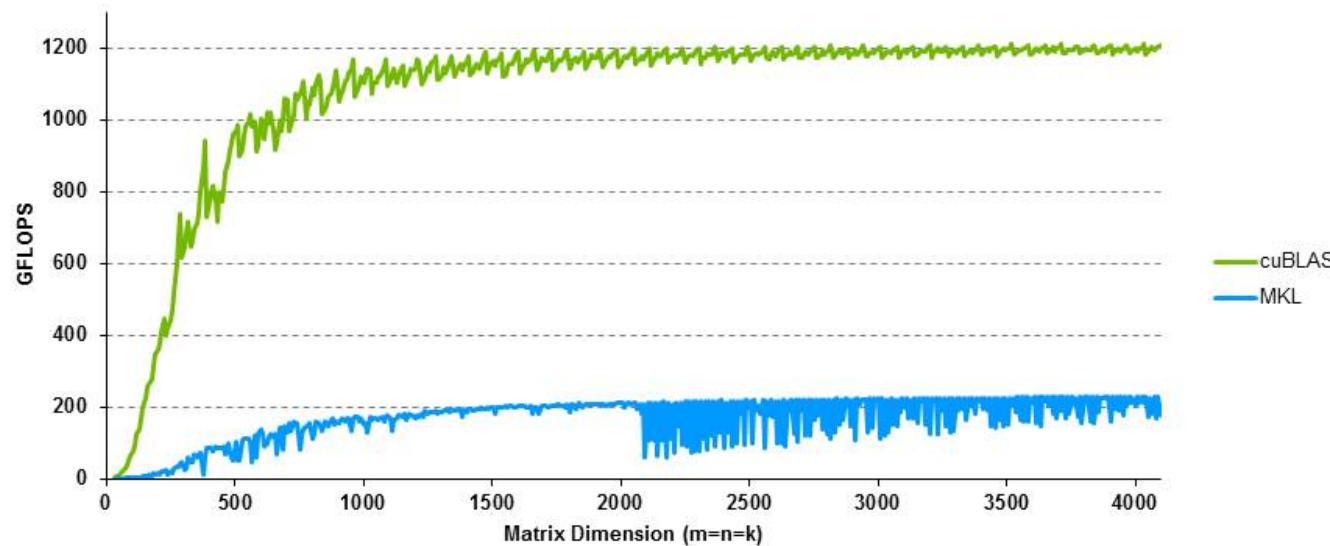


GPU Tesla C2050 (Fermi):
448 CUDA cores @ 1.15GHz
Double precision peak is 515 GFlop/s
[system cost ~\$3,000]

CPU AMD ISTANBUL
8 socket 6 core (48 cores) @ 2.8GHz
Double precision peak is 538 GFlop/s
[system cost ~\$30,000]

MAGMA: Matrix Algebra on GPU and Multicore Architectures

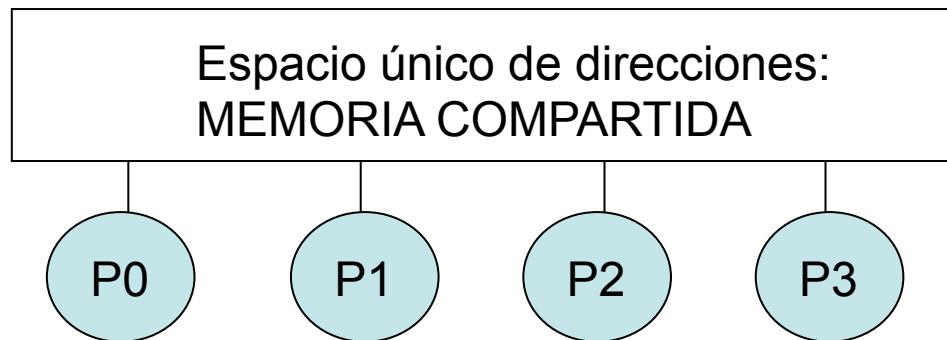
cuBLAS: ZGEMM 5x Faster than MKL



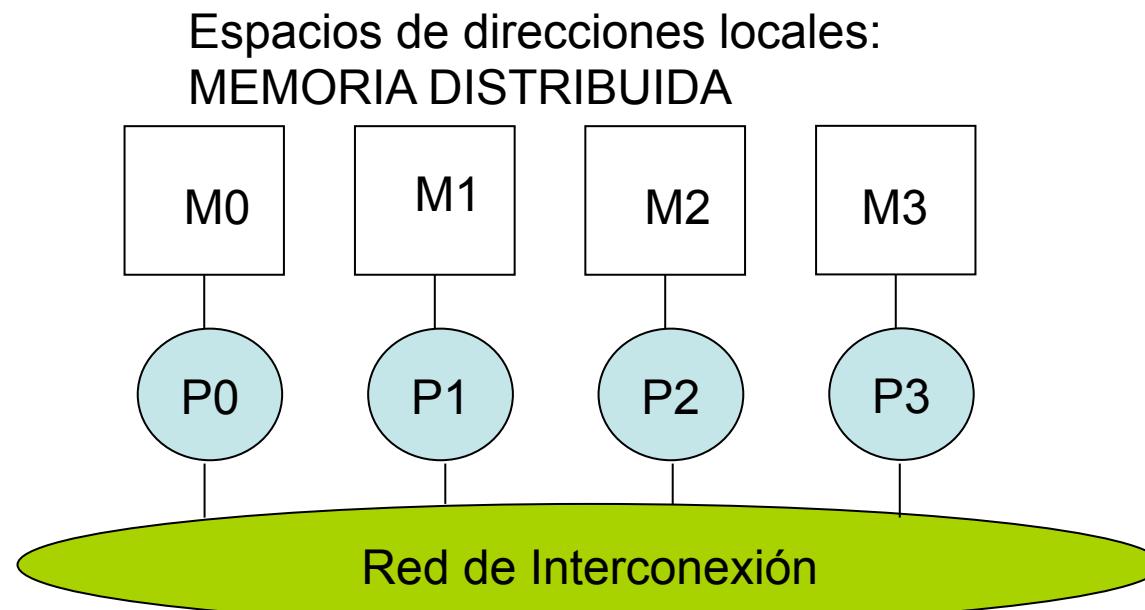
Performance may vary based on OS version and motherboard configuration

- cuBLAS 6.0 on K40m, ECC ON, input and output data on device
- MKL 11.0.4 on Intel IvyBridge 12-core E5-2697 v2 @ 2.70GHz

Multiprocesadores y Multicomputadores



Hacia 1990 empiezan a tomar protagonismo los multiprocesadores con memoria distribuida o multicomputadores



Entornos de paso de mensajes

- Aparecen los entornos de paso de mensajes que facilitan la programación de estas máquinas
 - PVM (Parallel Virtual Machine): 1989-1990
 - MPI (Message Passing Interface): 1994
- El entorno MPI se ha convertido en el estándar de facto del modelo de paso de mensajes
- Ofrece las primitivas de comunicación necesarias para implementar las comunicaciones
- Permiten el desarrollo software de herramientas eficientes de computación: Librerías Matriciales

Librerías para el modelo de paso de mensajes

La librería ScaLAPACK (Scalable Linear Algebra Package) (1995)

- Librería de rutinas de álgebra lineal numérica, para ordenadores de memoria distribuida con paso de mensajes.
- Capacidades similares al LAPACK: Sistemas lineales, Mínimos cuadrados, Valores y vectores propios y singulares, Descomposiciones matriciales
- Objetivos:
 - Eficiencia
 - Escalabilidad
 - Fiabilidad
 - Portabilidad
 - Flexibilidad
 - Facilidad de uso

La librería Scalapack. Filosofía y entorno.

Se trata de emplear la misma filosofía de diseño que en el LAPACK:

Núcleos computacionales básicos

Operaciones vector-vector
Operaciones matriz-vector
Operaciones matriz-matriz



Rutinas de alto nivel

Resolución de sistemas
Cálculo de valores propios
Cálculo de valores singulares
Descomposiciones matriciales

Problema: Hay que diseñar algoritmos distribuidos para los núcleos computacionales básicos, por ello es necesario un nivel más dedicado a las comunicaciones específicas que se utilizan en estos programas

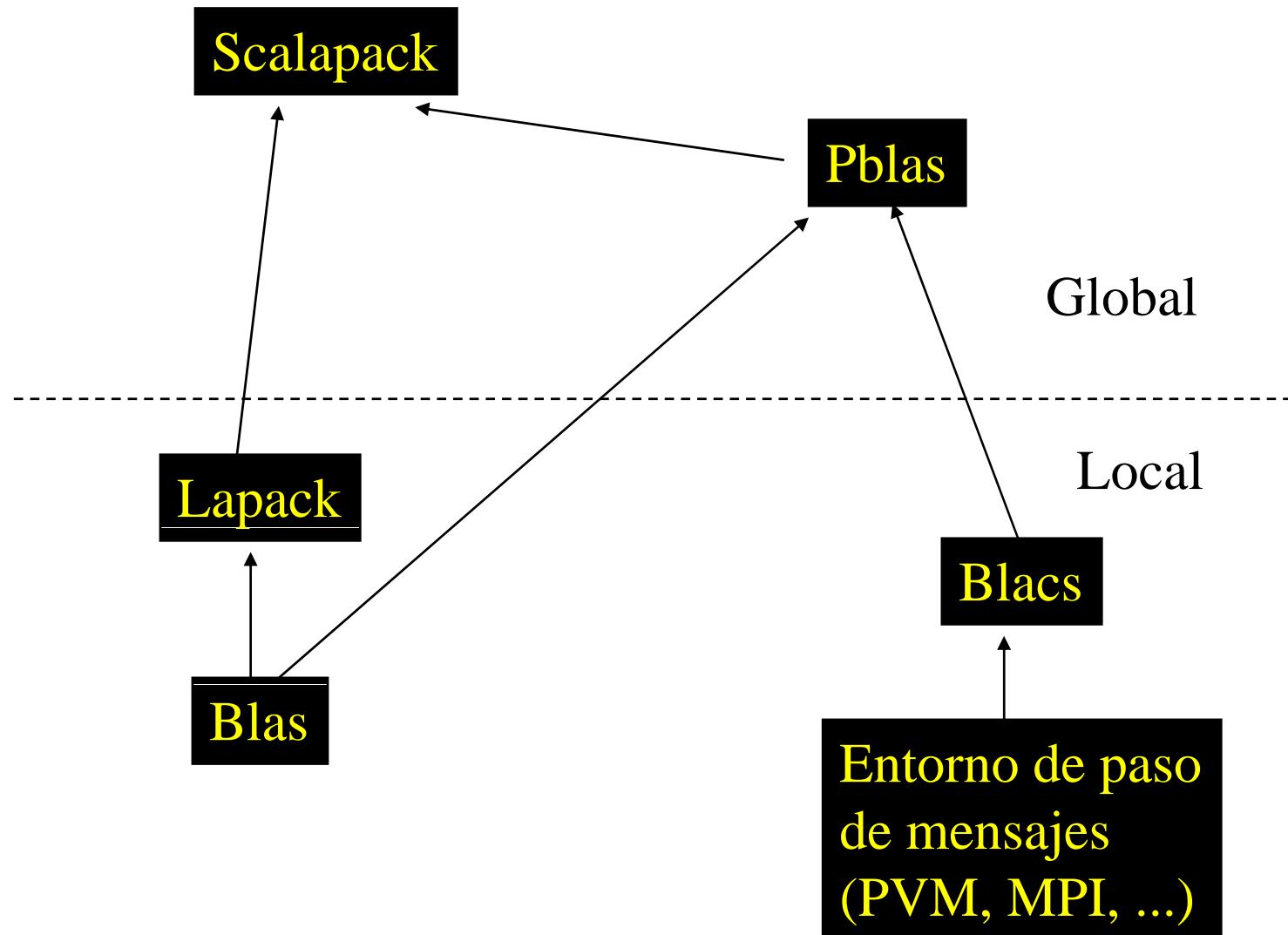
BLACS → **PBLAS** → **ScaLAPACK**

**Basic Linear
Algebra
Communication
Subprograms**

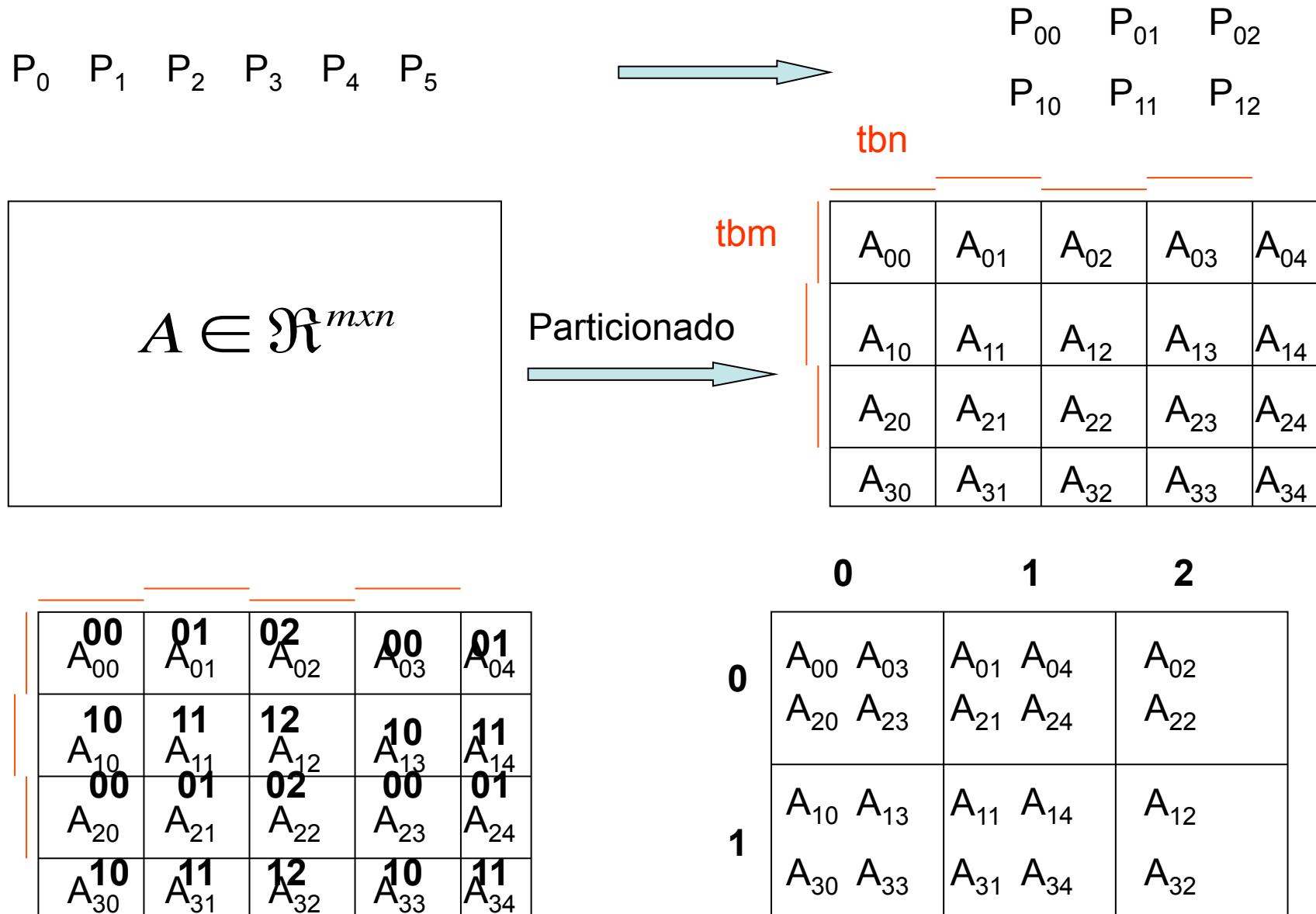
**Parallel Basic
Linear Algebra
Subprograms**

Scalable LAPACK

Scalapack



Distribución cíclica 2D: Distribución ScaLAPACK



SCALAPACK

Nomenclatura de las rutinas

Básicamente igual al LAPACK
formato Pxyyzz

<i>x</i> (<i>Tipo de datos</i>)	<i>yy</i> (<i>Tipo de matrices</i>)
S Reales	GE Rectangulares generales
D Doble Precisión	HE Hermitianas
C Complejos	SY Simétricas
Z Complejos dobles	TR Triangulares ...
<i>zz</i> (<i>Operación</i>)	

Clasificación: { Subrutinas driver
Subrutinas computacionales
Subrutinas auxiliares

SCALAPACK

Subrutinas Driver

- Rutinas driver
 - ❖ Resuelven un problema completo
 - ❖ Sistemas múltiples de ecuaciones lineales: $AX=B$: **PDGESV**
 - ❖ Problemas de mínimos cuadrados: $\min_x \|Ax-b\|^2$: **PDGELS**
 - ❖ Cálculo de valores propios: $Ax=\lambda x$: **PDSYEV**
 - ❖ Descomposición en valores singulares: $A=USV^T$: **PDSYEV**

BLACS

- Librería de comunicaciones para aplicaciones de álgebra lineal, a ejecutarse en multiprocesadores con paso de mensajes.
- Puede funcionar indistintamente sobre PVM, MPI u otras librerías de paso de mensajes.
- Fácil de programar, portable (Clusters de PSs/Workstations, Thinking Machine CM-5, IBM SP series, Cray T3X, Intel IPSC2, IPSC/860, Delta, Paragon, ...)

BLACS: Malla de Procesos

Dados N procesos (0, ..., N-1), generados con MPI o PVM, se distribuyen en una malla bidimensional:

	0	1	2	3
0	0	1	2	3
1	4	5	6	7

Los procesos se referenciarán por sus coordenadas en la malla:
 $P=R*C$ procesos se mapean en una malla $R \times C$

BLACS: contextos

- Contexto en BLACS ≈ comunicador en MPI; habitualmente, malla equivaldrá a proceso
- Como en MPI, es un mecanismo para diseñar software “seguro”.
- Permiten
 - 1) Crear grupos de procesos
 - 2) Crear mallas solapadas y/o disjuntas.
 - 3) Aisljar mallas para que no interfieran con otras.

BLACS: Ejemplo (1)

```
// Obtener número de procesos //
CALL BLACS_PINFO(iam, nprocs)

// Si estamos en pvm, crear máquina virtual //
IF (nprocs.LT.1) THEN
    nprocs=...
    CALL BLACS_SETUP(iam, nprocs)
ENDIF

//Establecer dimensiones de malla de procesos//
nrows=...
ncols=...

//Obtener contexto por defecto y crear la malla//
CALL BLACS_GET(-1,0,ctxt)
CALL BLACS_GRIDINIT(ctxt, 'Row_major',nrows, ncols)
```

BLACS: Ejemplo

```
CALL BLACS_GRIDINFO(ctxt,nrows,ncols,myrow, mycol)
```

```
// Si el proceso no está en la malla, terminar //
```

```
    IF (myrow.eq.-1) GOTO 10
```

```
...
```

```
//realizar el trabajo //
```

```
...
```

```
CALL BLACS_GRIDEXIT(ctxt)
```

```
10
```

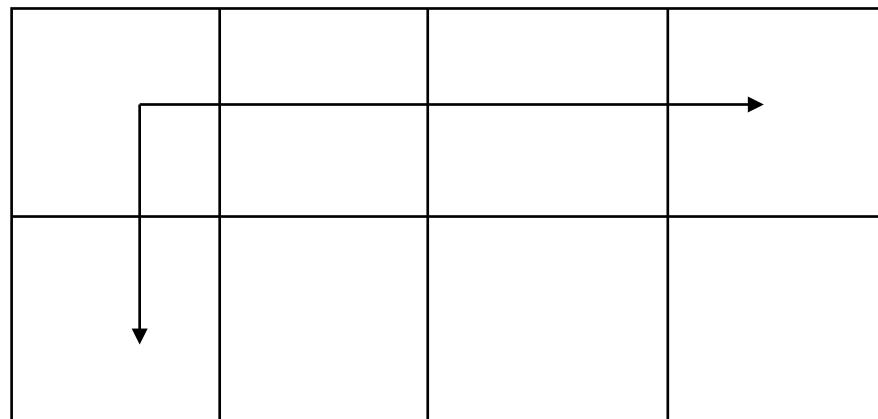
```
    CALL BLACS_EXIT(0)
```

BLACS: operaciones en ámbitos.

Operaciones en las que participa un grupo de procesos determinados.

Dada una malla de procesos bidimensionales, los ámbitos naturales son:

- fila
- columna
- malla



BLACS: subrutinas de comunicación punto a punto (1)

Envío / Recepción

`_xxSD2D(CTXT, [UPLO, DIAG], M, N, A, LDA, RDEST, CDEST)`

`_xxRV2D(CTXT, [UPLO, DIAG], M, N, A, LDA, RSRC, CSRC)`

<code>_</code> (<i>Tipo de datos</i>)	<code>xx</code> (<i>Tipo de matrices</i>)
I Enteros S, D Reales SP/DP C, Z Complejos SP/DP	GE Rectangulares generales TR Trapezoidales

- Envío de una (sub)matriz de un proceso a otro
- Envíos no bloqueantes
- Recepciones bloqueantes.

BLACS: subrutinas de comunicación punto a punto (2)

Ejemplo:

```
CALL BLACS_GRIDINFO(ctxt,nrows,ncols,myrow, mycol)
```

```
if (myrow.eq.0).and.(mycol.eq.0) then
    call dgesd2d(ctxt, 5, 1, X, 5, 1, 0)
    call dgerv2d(ctxt, 5, 1, X, 5, 1, 0)
else if (myrow.eq.1).and.(mycol.eq.0) then
    call dgesd2d(ctxt, 5, 1, X, 5, 0, 0)
    call dgerv2d(ctxt, 5, 1, X, 5, 0, 0)
end if
```

BLACS: subrutinas de difusión (broadcast) (1)

Envío / Recepción

`_xxBS2D(CTXT, SCOPE, TOP,[UPLO, DIAG], M, N, A, LDA)`

`_xxBR2D(CTXT,SCOPE, TOP,[UPLO, DIAG], M, N, A, LDA)`

<i>SCOPE</i> (ámbito)	<i>TOP</i> (topología)
'Row'	'` por defecto'
'Column'	'Increasing Ring'
'All'	'l-tree'

- xx (Tipo de matrices)
- Envío de una (sub)matriz a un grupo de procesos
- Operación globalmente bloqueante.

BLACS: subrutinas de difusión (broadcast) (2)

Ejemplo:

```
CALL BLACS_GRIDINFO(ctxt,nrows,ncols,myrow, mycol)
```

```
if (mycol.eq.2) then
    if (myrow.eq.0) then
        call dgebs2d(ctxt, 'COLUMN', ' ', 5, 7, B(9,4), 500)
    else if (myrow.eq.1)
        call dgebr2d(ctxt, 'COLUMN', ' ', 5, 7, WORK, 5, 0, 2)
    else
        call dgebr2d(ctxt, 'COLUMN', ' ', 5, 7, B(9,4), 500,0,2)
end if
```

BLACS: subrutinas de reducción

Suma/Máximo/Mínimo

`_GSUM2D(CTXT,SCOPE,TOP,M,N,A,LDA, RDEST,CDEST)`

`_GMAX2D(CTXT,SCOPE,TOP,M,N,A,LDA,RA,CA,RCFLAG,RDEST,CDEST)`

`_GMIN2D(CTXT,SCOPE,TOP,M,N,A,LDA,RA,CA,RCFLAG,RDEST,CDEST)`

BLACS: subrutinas de soporte(I)

→ Inicialización de la malla

BLACS_PINFO(IAM, NPROCS)

BLACS_SETUP(IAM, NPROCS)

BLACS_GRIDINIT(CTXT, ORDER, NROW, NCOL)

BLACS_GRIDMAP(CTXT, USERMAP, LDUMAP, NROW, NCOL)

→ Destrucción de la malla

BLACS_GRIDEXIT(CTXT)

BLACS_EXIT(CONTINUE)

BLACS_ABORT(CTXT, ENUM)

BLACS: subrutinas de soporte (II)

→ Información

BLACS_GRIDINFO(CTXT, NROW, NCOL, ROW, COL2)

BLACS_PNUM(CTXT, ROW, COL)

BLACS_PCOORD(CTXT, PNUM, ROW, COL)

BLACS_GET(CTXT, WHAT, VAL)

BLACS_BARRIER(CTXT, SCOPE)

* ~~Get system information~~ (out) uniquely identifies each process
(out) number of processes available
~~CALL BLACS_PINFO(IAM, NPROCS)~~ (in) integer handle indicating the context

* ~~Get default system context~~ (in) use (default) system context
(out) BLACS context
~~CALL BLACS_GET(0, 0, ICTXT)~~

* ~~Define 1 x (NPROCS/2+1) process grid~~ (output)
process row and column coordinate
 NPROW = 1
 NPCOL = NPROCS / 2 + 1
 CALL BLACS_GRIDINIT(ICTXT, 'Row', NPROW, NPCOL)
 CALL BLACS_GRIDINFO(ICTXT, NPROW, NPCOL, MYROW, MYCOL)

* If I'm not in the grid, go to end of program
 IF(MYROW.NE.-1) THEN
 IF(MYROW.EQ.0 .AND. MYCOL.EQ.0) THEN
 CALL DGESD2D(ICTXT, 5, 1, X, 5, 1, 0) send X to process (1,0)
 ELSE IF(MYROW.EQ.1 .AND. MYCOL.EQ.0) THEN
 CALL DGERV2D(ICTXT, 5, 1, Y, 5, 0, 0)
 END IF
 CALL BLACS_GRIDEXIT(ICTXT) receive X from process (0,0)
 END IF
 CALL BLACS_EXIT(0) leave context
 END

PBLAS

(Parallel Basic Linear Algebra Subprograms)

Introducción

- Librería de rutinas que implementan versiones paralelas de las rutinas del BLAS para memoria distribuida con paso de mensajes
- Objetivos:
 - Portabilidad
 - Altas prestaciones y escalabilidad
 - Facilidad de uso

PBLAS

Parallel Basic Linear Algebra Subprograms

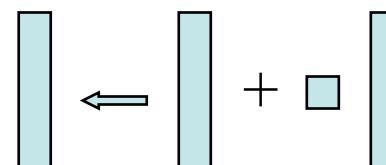
Librería de rutinas que implementan versiones paralelas de las rutinas del BLAS para memoria distribuida con paso de mensajes

Objetivos:

- Portabilidad
- Altas prestaciones y escalabilidad
- Facilidad de uso

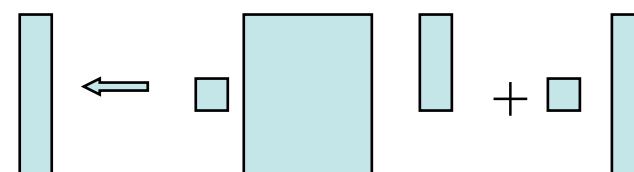
- Nivel 1: operaciones Vector-Vector

$$y \leftarrow \alpha x + y$$



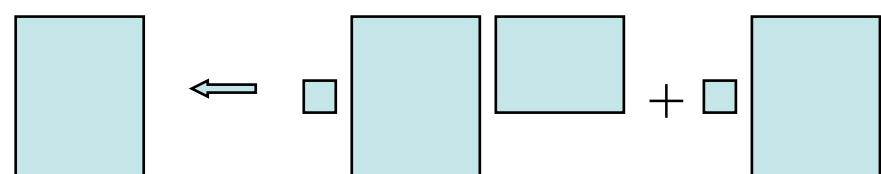
- Nivel 2: operaciones Matriz-Vector

$$\leftarrow \alpha A x + \beta y$$



- Nivel 3: operaciones Matriz-Matriz

$$C \leftarrow \alpha A B + \beta C$$



PBLAS

Nomenclatura de las rutinas

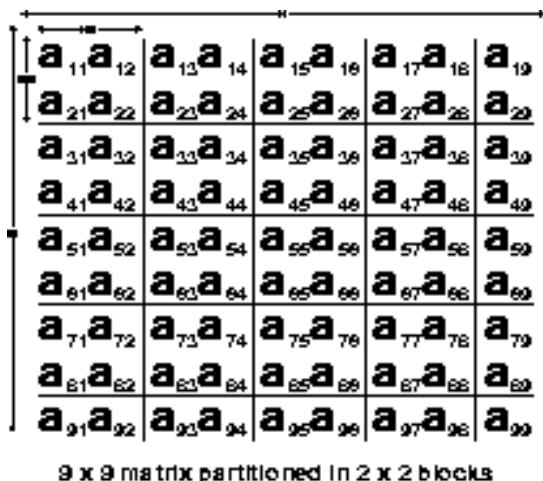
x (<i>Tipo de datos</i>)	yy (<i>Tipo de matrices</i>)
S Reales	GE Rectangulares generales
D Doble Precisión	HE Hermitianas
C Complejos	SY Simétricas
Z Complejos dobles	TR Triangulares (otros en BLAS 2)
zz (<i>Operación</i>)	
MV: producto matriz por vector	
MM: producto matriz por matriz	
SV, SM: resolución de sistemas triangulares	
R, R2, RK, R2K: actualizaciones de rango 1, rango 2, ...	

La rutina de transposición de matrices es P_TRAN_

PBLAS

Distribución de datos (1)

Distribución bidimensional cíclica por bloques



	0	1	2	
0	$a_{11}a_{12}$ $a_{21}a_{22}$ $a_{31}a_{32}$ $a_{41}a_{42}$ $a_{51}a_{52}$ $a_{61}a_{62}$ $a_{71}a_{72}$ $a_{81}a_{82}$ $a_{91}a_{92}$	$a_{17}a_{18}$ $a_{27}a_{28}$ $a_{37}a_{38}$ $a_{47}a_{48}$ $a_{57}a_{58}$ $a_{67}a_{68}$ $a_{77}a_{78}$ $a_{87}a_{88}$ $a_{97}a_{98}$	$a_{13}a_{14}$ $a_{23}a_{24}$ $a_{33}a_{34}$ $a_{43}a_{44}$ $a_{53}a_{54}$ $a_{63}a_{64}$ $a_{73}a_{74}$ $a_{83}a_{84}$ $a_{93}a_{94}$	$a_{19}a_{20}$ $a_{29}a_{30}$ $a_{39}a_{40}$ $a_{49}a_{50}$ $a_{59}a_{60}$ $a_{69}a_{70}$ $a_{79}a_{80}$ $a_{89}a_{90}$ $a_{99}a_{100}$
1	$a_{31}a_{32}$ $a_{41}a_{42}$ $a_{51}a_{52}$ $a_{61}a_{62}$ $a_{71}a_{72}$ $a_{81}a_{82}$ $a_{91}a_{92}$	$a_{37}a_{38}$ $a_{47}a_{48}$ $a_{57}a_{58}$ $a_{67}a_{68}$ $a_{77}a_{78}$ $a_{87}a_{88}$ $a_{97}a_{98}$	$a_{33}a_{34}$ $a_{43}a_{44}$ $a_{53}a_{54}$ $a_{63}a_{64}$ $a_{73}a_{74}$ $a_{83}a_{84}$ $a_{93}a_{94}$	$a_{39}a_{40}$ $a_{49}a_{50}$ $a_{59}a_{60}$ $a_{69}a_{70}$ $a_{79}a_{80}$ $a_{89}a_{90}$ $a_{99}a_{100}$

2 x 3 process grid point of view

PBLAS

Distribución de datos(2)

Propiedades:

- Engloba muchas otras distribuciones.
- Permite obtener carga balanceada
- Permite obtener altas prestaciones y escalabilidad

Es responsabilidad del programador distribuir las matrices entre todos los procesos antes de llamar a las subrutinas de PBLAS que operan con ellas.

Cada proceso tiene sus propios bloques que se almacenan en memoria local en columnas.

PBLAS

Distribución de datos(3)

Descriptor de array:

Todos los parámetros que describen una matriz distribuida se encapsulan en un descriptor de matriz (array de enteros)

DESC_()	Symbolic Name	Scope	Definition
1	DTYPE_A	(global)	Descriptor type DTTYPE_A=1 for dense matrices.
2	CTXT_A	(global)	BLACS context handle, indicating the BLACS process grid over which the global matrix A is distributed. The context itself is global, but the handle (the integer value) may vary.
3	M_A	(global)	Number of rows in the global array A.
4	N_A	(global)	Number of columns in the global array A.
5	MB_A	(global)	Blocking factor used to distribute the rows of the array.
6	NB_A	(global)	Blocking factor used to distribute the columns of the array.
7	RSRC_A	(global)	Process row over which the first row of the array A is distributed.
8	CSRC_A	(global)	Process column over which the first column of the array A is distributed.
9	LLD_A	(local)	Leading dimension of the local array. LLD_A $\geq \text{MAX}(1, \text{LOC}_r(M_A))$.

PBLAS

Argumentos de las subrutinas

Las llamadas son prácticamente igual a las de BLAS:

LLAMADA BLAS:

CALL DGEMM(‘No transpose’, ‘No transpose’, M,K,N,ONE,
A(IA,JA), LDA, LDA, B(IB,JB), LDB, ZERO, C(IC,JC),
LDC)

LLAMADA PBLAS:

CALL PDGEMV(‘No transpose’, ‘No transpose’, M,K,N,
ONE, A, IA, JA, DESCA, B, IB, JB, DESC B, ZERO, C, IC JC,
DESCC)

PBLAS

Eficiencia y Escalabilidad

Es posible usar técnicas de solapado de computación y comunicación.

Tambien hay subrutinas que asignana una topología a un contexto:

PTOPSET(CTXT, OP, SCOPE, TOP)

PTOPGET(CTXT, OP, SCOPE, TOP)

Ej: CALL PTOPSET(CTXT, ‘Broadcast’, ‘Row, ‘S-ring’)

PBLAS

Conclusiones

- PBLAS pretende facilitar la paralelización de código secuencial desarrollado con BLAS.
- Para facilitar su uso, se ha conservado el interfaz prácticamente igual
- Se utiliza BLACS para las comunicaciones (sobre MPI o PVM) habitualmente.
- La distribución de carga bidimensional cíclica por bloques asegura buen balance de carga y garantiza altas prestaciones y escalabilidad.

SCALAPACK

Subrutinas Computacionales y Auxiliares

❑ Rutinas computacionales

- Resuelven diferentes tareas computacionales
- Algunas rutinas dependen de la salida de otras
- Factorizaciones LU, Cholesky, LDL^T , QR, ...
- Estimar nº de condición
- Cotas de error, ...

❑ Rutinas auxiliares

- Subtareas de algoritmos orientados a bloques, cálculos de bajo nivel, ...

SCALAPACK

Etapas para llamar a una rutina

- 1) Inicializar la malla de procesos (BLACS)
- 2) Distribuir la matriz en la malla
 - 2.1) Inicializar el descriptor de array
DESCINIT(DESCA,M,N,MB,NB,RSRC,CSRC,CTCX,LLD,INFO)
 - 2.2) Distribuir a cada proceso sus bloques locales
- 3) Llamar a la rutina
- 4) Liberar la malla de procesos (BLACS)

SCALAPACK

Prestaciones

Factores que pueden afectar a las prestaciones:

- Topología
- Tamaño de bloque
- Dimensiones de la malla de procesos
- Su eficiencia depende de las implementaciones de BLACS y PBLAS

Para un problema determinado es necesario estudiar los valores óptimos de estos parámetros

Estado actual: Existen implementaciones buenas pero su utilización se hace complicada. No se han desarrollado todas las rutinas que tiene el LAPACK.

SCALAPACK

Prestaciones

Para obtener buenas prestaciones en multicomputadores
(recomendaciones antiguas):

- Use the right number of processors:

Rule of thumb: $p=M \times N / 1000000$ for a $M \times N$ matrix. This provides a local matrix of size approximately 1000 by 1000.

- Do not try to solve a small problem on too many processors.
- Do not exceed physical memory.
- Use an efficient data distribution.
- Block size (MB,NB=64)
- Square processor grid
- Use efficient machine-specific BLAS (not the Fortran 77 reference implementation BLAS) and BLACS

SCALAPACK

Prestaciones

Para obtener buenas prestaciones en redes de ordenadores:

- The bandwidth per node, in Mbytes/second/node, should be no less than one tenth of the peak floating-point rate, in Mflops/second/node.
- The underlying network must allow simultaneous messages (not standard ethernet)
- Message latency should be no more than 500 microseconds.
- All processors should be similar in architecture and performance. ScaLAPACK will be limited by the slowest processor. Data format conversion significantly reduces communication performance.
- No other jobs should be allowed to execute on the processors that are being used. If the processors are gang scheduled and there is enough physical memory for all jobs on all processors, this requirement may be relaxed, but we do not recommend doing so without careful study.
- No more than one process should be executed per processor.

SCALAPACK

Prestaciones

Para mejorar las prestaciones:

- Use the best BLAS and BLACS libraries available.
- Start with a standard data distribution.
 - A square processor grid ($r=c=\sqrt{p}$) if $p>9$
 - A one dimensional processor grid ($r=1,c=p$) if $p<9$
 - Block size = 64
- Determine whether reasonable performance is being achieved.
- Identify the performance bottleneck(s), if any,
- Tune the distribution or routine parameters to improve performance further.

SCALAPACK

Conclusiones

- Scalapack es la versión de LAPACK para ordenadores con memoria distribuida.
- Librerías previas: (MPI o PVM), BLACS, PBLAS, BLAS, BLACS, LAPACK.
- Su eficiencia depende de las implementaciones de BLACS y BLAS y de la arquitectura del multicomputador
- Estado actual:
 - Existen implementaciones buenas pero su utilización se hace complicada.
 - No se han desarrollado todas las rutinas que tiene el LAPACK.
- Estado futuro:
 - ¿Tiene futuro ScaLAPACK?

Referencias

Referencias BLACS:

- Página principal: <http://www.netlib.org/blacs/>
- LAPACK Working Note 94, "A User's Guide to the BLACS v1.1", Jack J. Dongarra, R. Clint Whaley, May 5, 1997

Referencias PBLAS:

- Online quick reference guide:
http://www.netlib.org/scalapack/html/pblas_qref.html

Referencias ScaLAPACK:

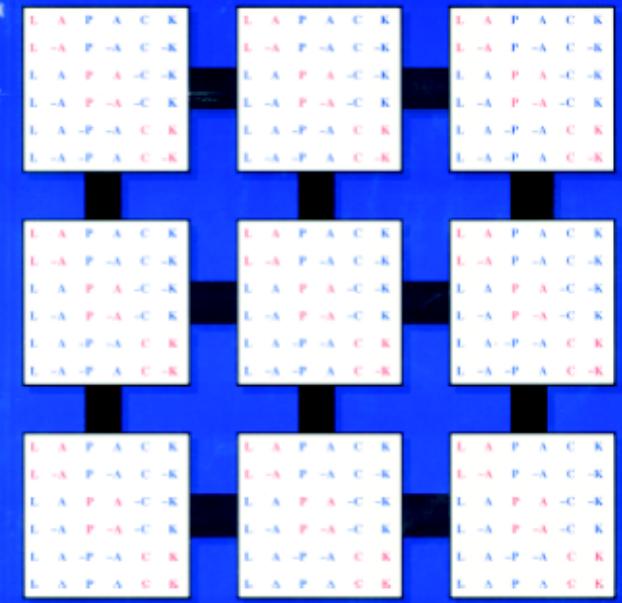
- Página principal:
http://www.netlib.org/scalapack/scalapack_home.html
- User's guide v1.7:
<http://www.netlib.org/scalapack/slug/index.html>

Información sobre ScaLAPACK

<http://www.netlib.org/scalapack/slug/index.html>

ScaLAPACK Users' Guide

L. S. Blackford • J. Choi • A. Cleary • E. D'Azevedo
J. Demmel • I. Dhillon • J. Dongarra • S. Hammarling
G. Henry • A. Petitet • K. Stanley • D. Walker • R. C. Whaley



Team of Developers:

- Susan Blackford
- Jaeyoung Choi, Soongsil University
- Andy Cleary, LLNL
- Ed D'Azevedo, ORNL
- Jim Demmel, UCB
- Inderjit Dhillon, UT Austin
- Jack Dongarra, UTK
- Ray Fellers, LLNL
- Sven Hammarling, NAG
- Greg Henry, Intel
- Sherry Li, LBNL
- Osni Marques, LBNL
- Caroline Papadopoulos, UCSD
- Antoine Petitet, UTK
- Ken Stanley, UCB
- Francoise Tisseur, Manchester

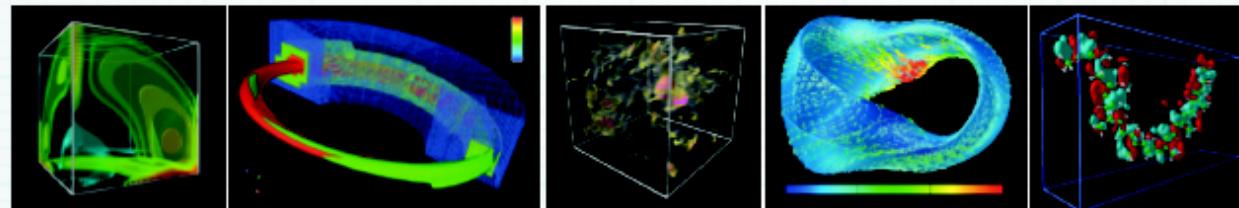
La colección ACTS

<http://acts.nersc.gov>

- The Advanced CompuTational Software (ACTS) Collection is a **set of software tools for computational sciences**. The starting point for the collection was the former Advanced Computational Testing and Simulation Toolkit Project. The purpose of the ACTS Collection is to **accelerate the adoption and use of advanced computing by the Department of Energy programs for their mission-critical problems**. While DOE has been motivated to develop the tools for its own programs, it also encourages their adoption and use by non-DOE computational efforts. The ACTS Collection Project disseminates information about the tools, both inside and outside the DOE community. In addition, it focuses on the installation, maintenance and support of the tools on DOE's computing facilities.

La colección ACTS

The U.S. DOE ACTS Collection

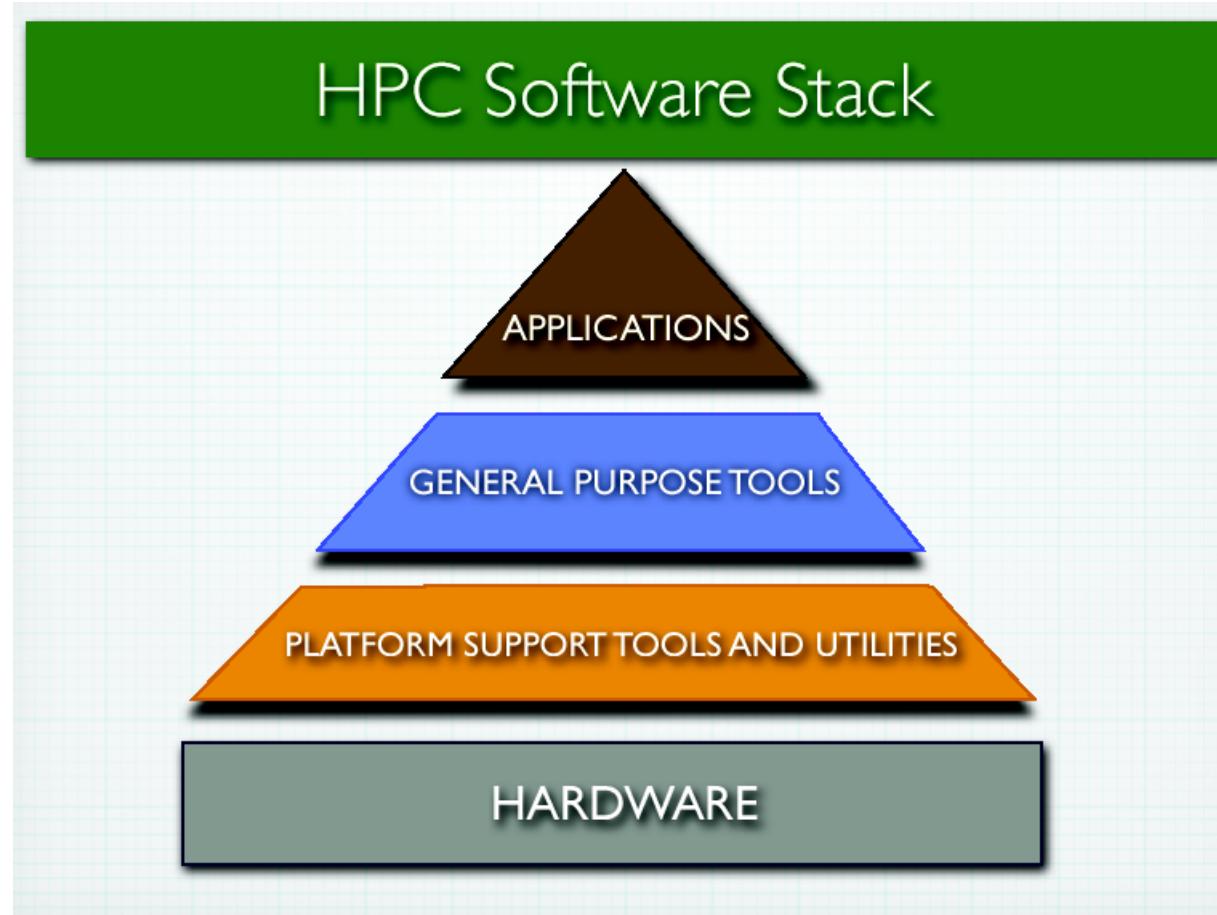


Goal: The Advanced Computational Software Collection (ACTS) makes reliable and efficient software tools more widely used, and more effective in solving the nation's engineering and scientific problems.

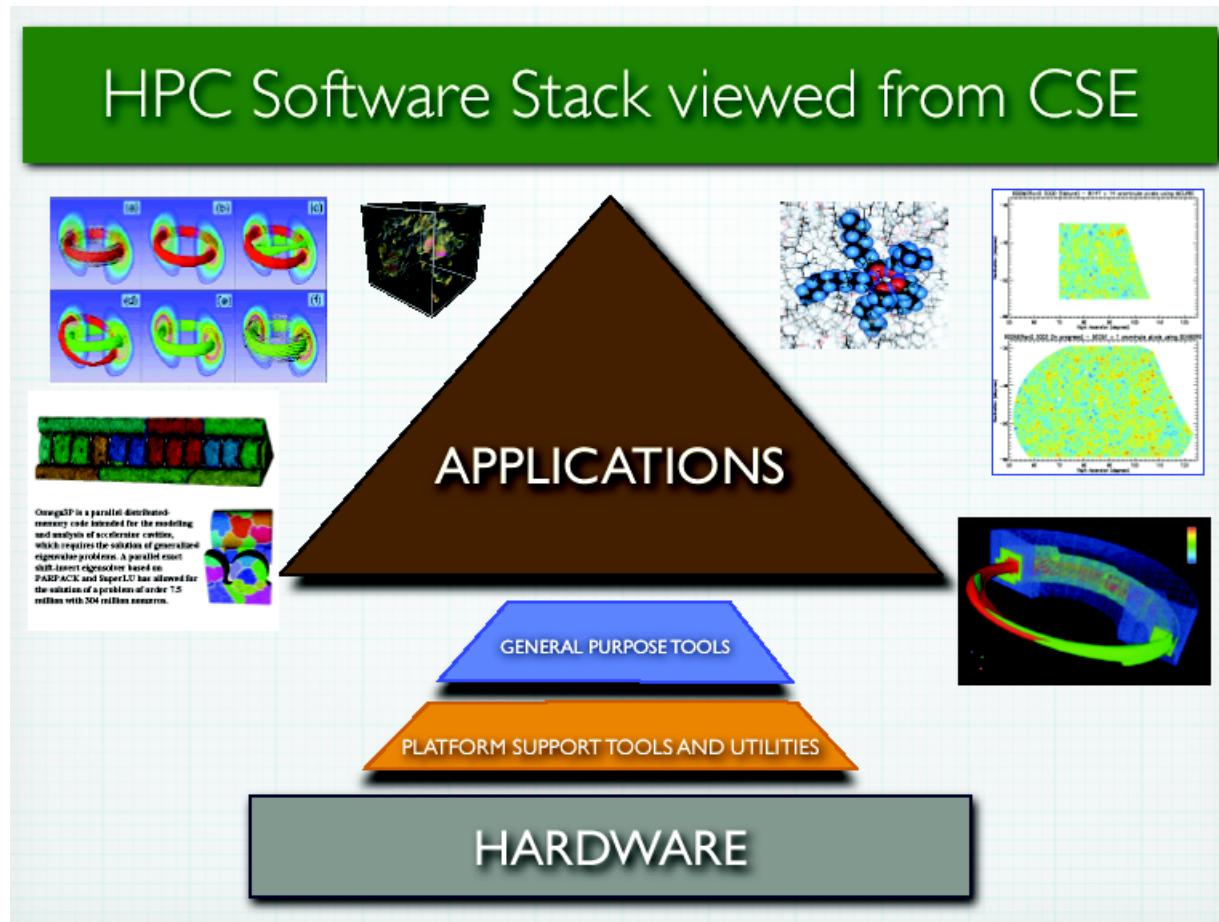
References:

- L.A. Drummond, O. Marques: An Overview of the Advanced Computational Software (ACTS) Collection. ACM Transactions on Mathematical Software Vol. 31 pp. 282-301, 2005
- <http://acts.nersc.gov>

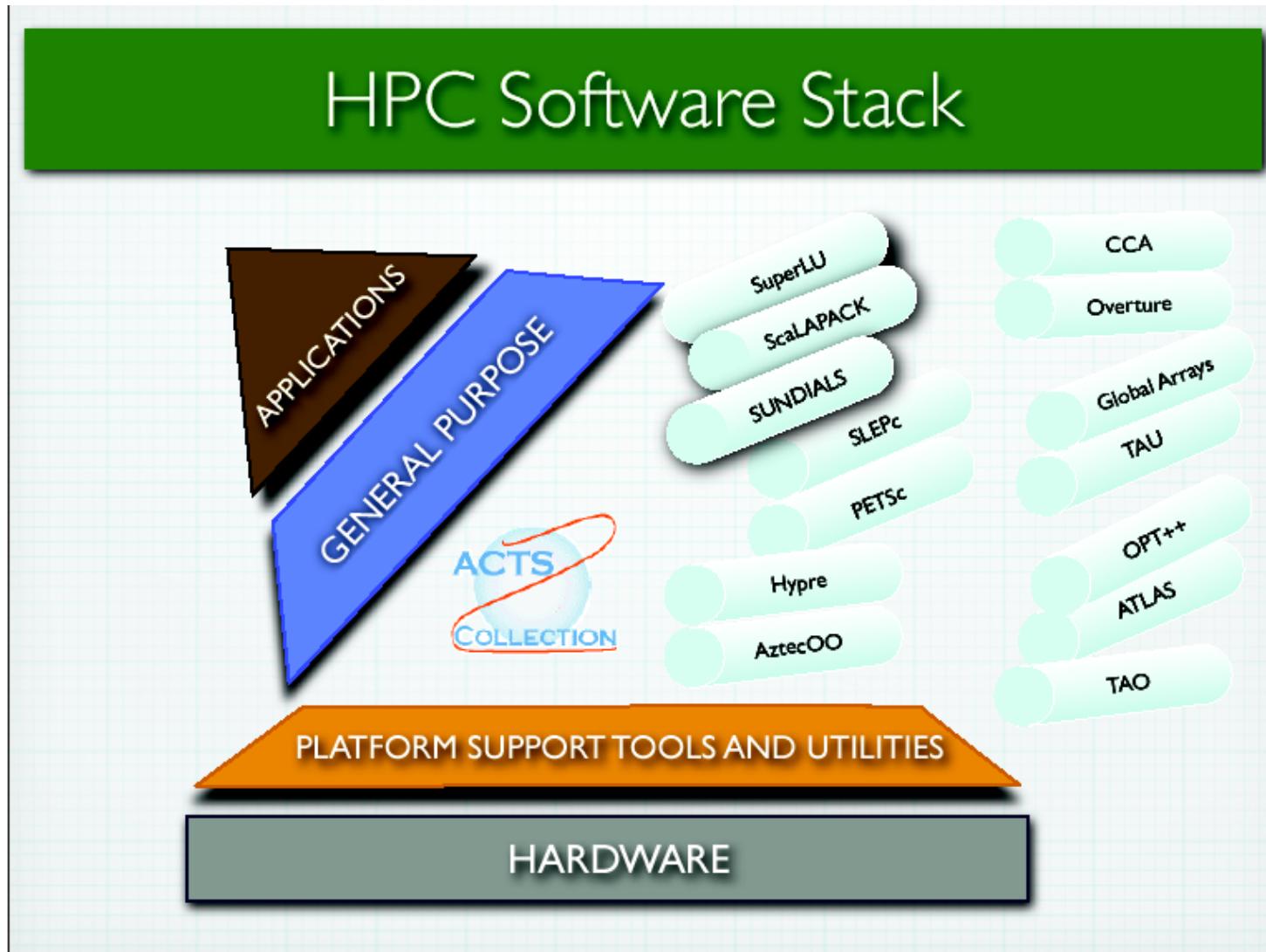
La colección ACTS



La colección ACTS



La colección ACTS



La colección ACTS

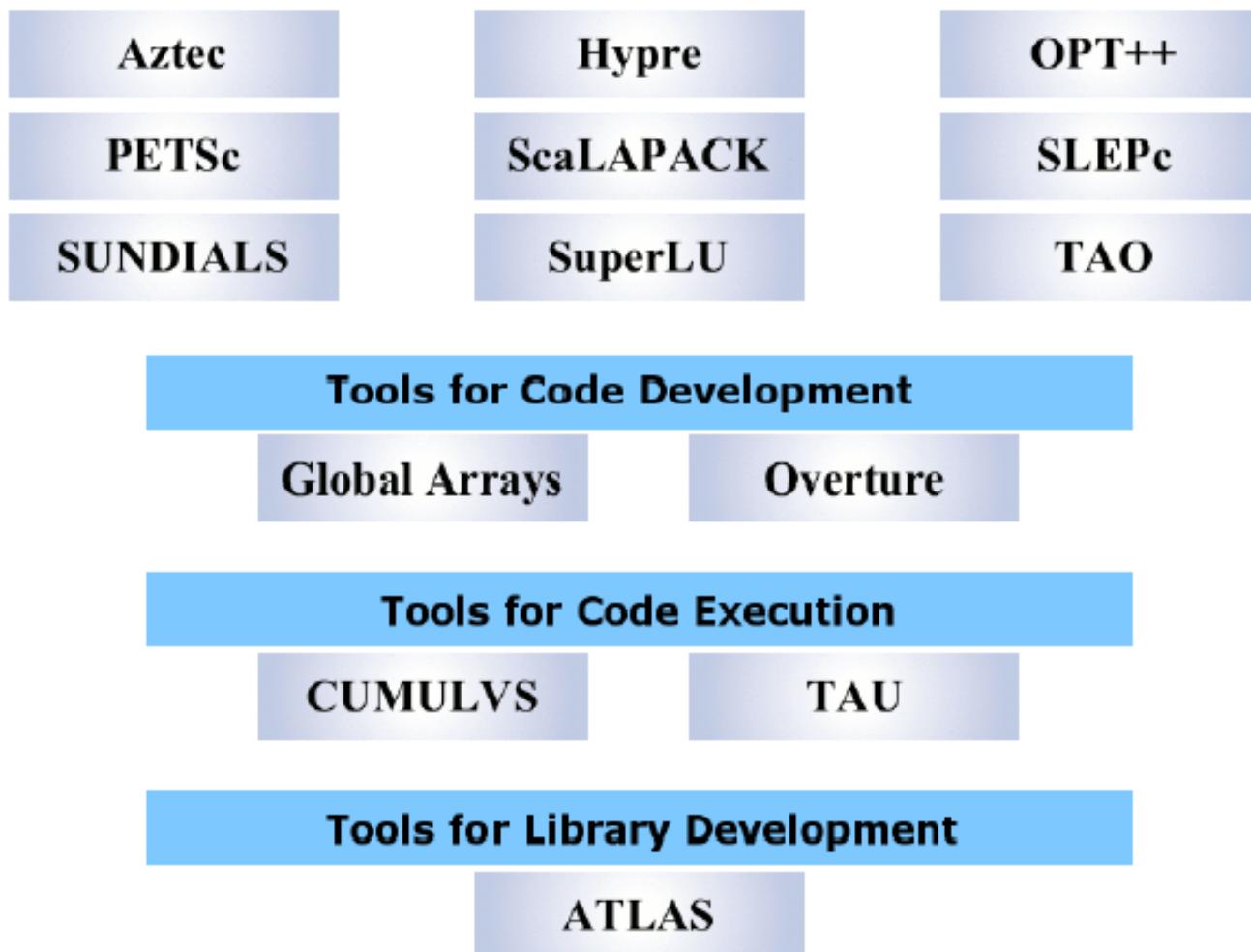
The U.S. DOE ACTS Collection

Category	Tool	Functionalities
Numerical	AztecOO	Scalable linear and non-linear solvers using iterative schemes.
	Hypre	A family of scalable preconditioners.
	PETSc	Scalable linear and non-linear solvers and additional support for PDE related work.
	OPT++	Object-oriented nonlinear optimization solvers.
	SUNDIALS	Solvers for the solution of systems of ordinary differential equations, nonlinear algebraic equations, and differential-algebraic equations.
	ScalAPACK	High performance parallel dense linear algebra.
	SLEPc	Scalable algorithms for the solution of large sparse eigenvalue problems.
	SuperLU	Scalable direct solution of large, sparse, nonsymmetric linear systems of equations.
	TAO	Large-scale optimization software.
Code Development	Global Arrays	Supports the development of parallel programs.
	Overture	Supports the development of computational fluid dynamics codes in complex geometries.
	CCA	Forum and a framework for Software Interoperability
Run Time Support	TAU	Portable and scalable performance analyzes and tracing tools for C, C++, Fortran and Java programs.
Library Development	ATLAS	Automatic generation of optimized numerical dense algebra for scalar processors.



La colección ACTS

<http://acts.nersc.gov/tools.html>



StructPack

- Biblioteca de Altas Prestaciones para la Resolución de Problemas Matriciales Estructurados

<http://www.inco2.upv.es/structpack.html>

E-mail: structpack@dsic.upv.es

Departamento de Sistemas Informáticos y Computación (Universidad Politécnica de Valencia)

Departamento de Informática (Universidad de Oviedo)

Departamento de Comunicaciones - ITEAM (Universidad Politécnica de Valencia)

- Resuelve sistemas de ecuaciones lineales y problemas de mínimos cuadrados sobre matrices estructuradas: Toeplitz, Hankel, Vandermonde,...

StructPack Library

[Home](#) [Installation](#) [Documentation](#) [Test](#) [FAQs](#)

High Performance Computing Library for structured Matrices

StructPack

StructPack is a collection of numerical routines written in Fortran90/95 for efficiently solving numerical Linear Algebra problems on structured matrices. It is designed for Linux environments, and optimized for use on sequential CPU type and multicore architectures; OpenMP API has been used in its development. It also provides executable programs for solving problems directly from the command line and .mex files for using the routines in Matlab and GNU Octave.

Working Notes

References

Third Party Software

New Releases

1th Jul 2012 • StructPack v1.2 released
Source code. New features, bug fixes.

1th Jun 2011 • StructPack v1.1 released
Source code. New features, bug fixes.

Contact us If you have any comments or questions regarding StructPack Library you will be contacted via e-mail.

Groups



(IRPCG)

Projects



PROMETEO: High Performance Computing Tools for solving Signal Processing Problems on Parallel Architectures. (PROMETEO/2009/013. Generalitat Valenciana. Spain.)

COPABIB: Automatic Development and Tuning of Parallel Libraries for Scientific Computation (TIN2008-06570-C04-02. Ministerio de Ciencia e Innovación. Spain).

PAID-UPV: Development and Implementation of Computational Kernels for Software-Defined Radio Platforms on Multicore / Manycore Architectures. (PAID-05-10).

RECAPAD: Characteristic Reduction and High Performance Computing in High Dimensionality Problems (TIN2010-14971, Ministerio de Ciencia e Innovación, Spain).

Routines included in StructPack



Currently, the routines included in StructPack can be used to: Solve symmetric, tridiagonal, Toeplitz, linear systems. Calculate eigensystems and singular value decomposition of symmetric tridiagonal Toeplitz matrices. Solve symmetric Toeplitz linear systems of equations. Solve non-symmetric Toeplitz linear systems.

StructPack v1.2 released



Source code. New features, bug fixes.

StructPack v1.1 released



Source code. New features, bug fixes.

StructPack Library

StructPack was conceived for solving a variety of numerical Linear Algebra problems, such as solving systems of linear equations, least squares problems, eigenvector calculation and singular value decomposition, on different types of matrices such as Toeplitz, Hankel, Vandermonde, or circulant matrices, ... and specific cases as tridiagonal Toeplitz matrices, positive definite symmetric matrices, etc.

[Working Notes](#)

[References](#)

[Third Party Software](#)

New Releases

1th Jul 2012 • StructPack v1.2 released
Source code. New features, bug fixes.

1th Jun 2011 • StructPack v1.1 released
Source code. New features, bug fixes.

Contact us If you have any comments or questions regarding StructPack Library you will be contacted via e-mail.

High Performance Computing Library for structured Matrices

StructPack

StructPack is a collection of numerical routines written in Fortran90/95 for efficiently solving numerical Linear Algebra problems on structured matrices. It is designed for Linux environments, and optimized for use on sequential CPU type and multicore architectures; OpenMP API has been used in its development. It also provides executable programs for solving problems directly from the command line and .mex files for using the routines in Matlab and GNU Octave.

SPAWNs (StructPack Working Notes)

SPAWN 001 [pdf]

StructPack: A high performance library for structured matrices.
by Pedro Alonso-Jordá, Pablo Martínez-Naredo, F.J. Martínez-Zaldivar, José Ranilla, Carla Ramiro and Antonio M. Vidal.
June 24,2011

Como se diseñan e implementan algoritmos secuenciales / paralelos para una librería tipo LAPACK / ScaLAPACK?

Trabajo matemático: elegir el mejor método

Trabajo computacional: analizar sus posibilidades computacionales en cuanto a prestaciones como eficiencia, robustez, precisión, análisis de error...

Trabajo de implementación: seguir una metodología de implementación precisa y contrastada

Ejemplo: Descomposición QR en LAPACK /ScaLAPACK

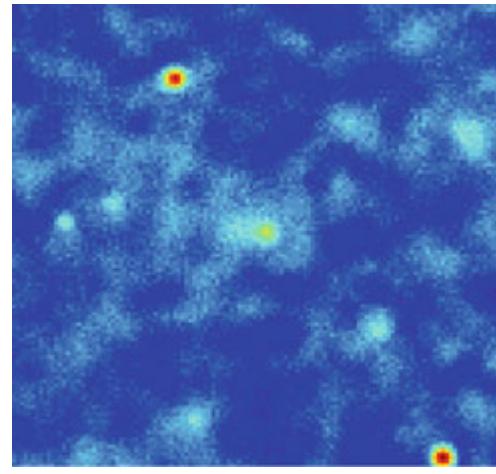
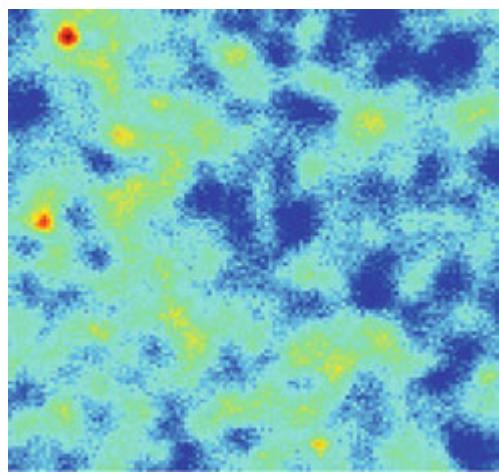
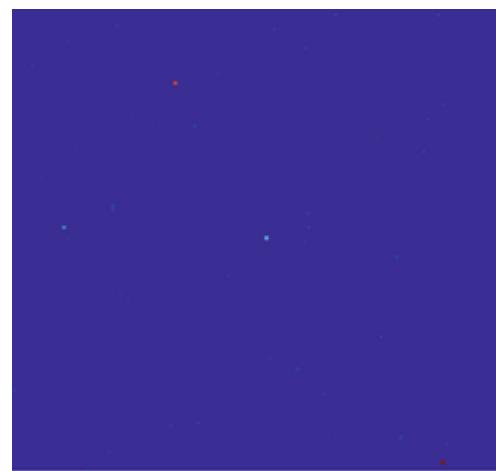
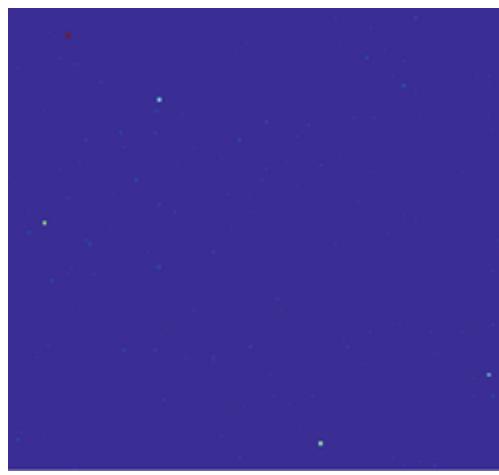
Caso de estudio: **Detecting point sources in CMB maps using HPC Libraries**

Referencias:

- “A Bayesian technique for the detection of point sources in CMB maps”.
Argüeso F., Salerno E., Herranz D., Sanz J.L., Kuruoglu E.E., Kayabol K.:
Mon. Not. Roy. Astron. Soc. 414, 410–417 (2011)
- “Detecting point sources in CMB maps using an efficient parallel algorithm”.
P.Alonso, F.Argüeso, R.Cortina, J.Ranilla and A.M.Vidal. Journal of
Mathematical Chemistry. Volume 50, Issue 2 , pp 410-420. (2012)
- “Non-linear parallel solver for detecting point sources in CMB maps using
Bayesian techniques”. P.Alonso, F.Argüeso, R.Cortina, J.Ranilla and
A.M.Vidal. Journal of Mathematical Chemistry. Published online 16 October
(2012).

El problema

- The **Cosmic Microwave Background (CMB)** is a diffuse radiation which is contaminated by the radiation emitted by point sources. The precise knowledge of CMB fluctuations can lead to a better knowledge of the chemistry at the early stages of the Universe. In this work, we present an efficient algorithm, with a high degree of parallelism, which can improve, from the computational point of view, the classical approaches for detecting point sources in Cosmic Microwave Background maps. **High performance computing libraries and parallel computing techniques have allowed to construct a portable, fast and numerically stable algorithm.** To check the performance of the new method, we have carried out several simulations resembling the observational data collected by the Low Frequency Instrument of the Planck satellite. The sources are detected in their real positions.



Modelado del problema

$$d(x, y) = \sum_{\alpha=1}^n a_\alpha b(x - x_\alpha, y - y_\alpha) + (f * b)(x, y) + n(x, y)$$

Incógnitas

Radiación observada Radiación de las fuentes Efecto de la CMB Ruido

Tras ciertas reordenaciones

$$d = \phi a + c, \text{ con } \phi \in \Re^{N \times n}, N \gg n$$

Información Incógnitas Ruido Gaussiano

Imponiendo las condiciones adecuadas para minimizar el ruido y haciendo un filtrado para calcular el valor de n , se llega al siguiente

Problema matricial

$$Ma = e \quad \text{con} \quad M = \phi^T \xi^{-1} \phi \quad \phi \in N \times n, \text{Def. Positiva}; \xi \in N \times N, N \gg n$$
$$e = \phi^T \xi^{-1} d$$

The diagram illustrates the transformation of the system $Ma = e$ into a matrix form. It shows three equations side-by-side. The first equation $Ma = e$ has an arrow pointing from its right side to the first term in the second equation. The second equation $M = \phi^T \xi^{-1} \phi$ has an arrow pointing from its left side to the first term in the third equation. The third equation $e = \phi^T \xi^{-1} d$ has an arrow pointing from its right side to the second term in the second equation. The terms M , a , e , ϕ , ξ , d , and ϕ^T are represented by light blue rectangular boxes, while the operators $=$ and \cdot are represented by vertical light blue lines.

Solución clásica

1. Calcular M
2. Calcular e
3. Resolver el sistema

Dificultades y soluciones

- Evitar operaciones numéricamente inestables como el cálculo de la inversa
- Utilizar moderadamente la memoria
- Conseguir un tiempo de ejecución razonable
- Organizar el algoritmo de tal forma que se puedan usar Librerías de Altas Prestaciones

Solución utilizando Librerías de Altas Prestaciones

Sea $\xi = L * L^T$, con L triangular inferior (Descomposición de Cholesky)

$$\begin{array}{ccc} \text{square matrix} & = & \text{lower triangular} \\ e = \phi^T \xi^{-1} d & = & \phi^T (L * L^T)^{-1} d = \phi^T L^{-T} L^{-1} d = \phi^T (L^{-T} (\underbrace{L^{-1} d}_{c_1})) = \phi^T c_2 \end{array}$$

$$M = \phi^T \xi^{-1} \phi = \phi^T (L * L^T)^{-1} \phi = (L^{-1} \phi)^T (L^{-1} \phi) = Z^T Z = (QR)^T (QR) = R^T (Q^T Q) R = R^T R$$

con $Z = QR$, (Desc.QR de Z ; Q ortogonal: $Q^T Q = I$)

Algoritmo

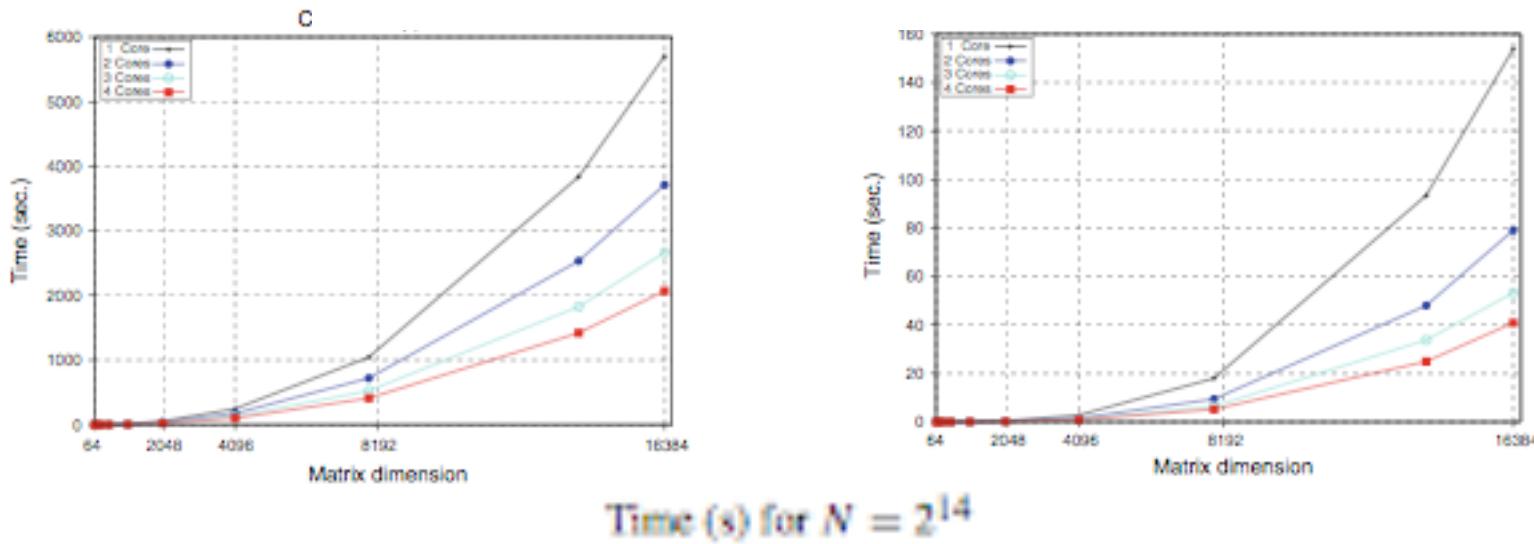
Resolver $Ma = e \Leftrightarrow R^T Ra = e \Leftrightarrow R^T y = e$ con $Ra = y$

1. Calcular la Descomposición de Cholesky: $\xi = L * L^T$ LAPACK
SUBROUTINE DPOTRF(UPLO, N, A, LDA, INFO)
2. Resolver el sistema de ecuaciones triangular inferior BLAS2
dtrsv (UPLO, TRANS, DIAG, N, A, LDA, X, INCX) $Lc_1 = d$
3. Resolver el sistema de ecuaciones triangular superior BLAS2
dtrsv (UPLO, TRANS, DIAG, N, A, LDA, X, INCX) $L^T c_2 = c_1$
4. Calcular el producto matriz-vector BLAS2
sgemv (TRANS, M, N, ALPHA, A, LDA, X, INCX, BETA, Y, INCY)
5. Resolver el sistema de ecuaciones triangular inferior con multiples términos independientes BLAS3
dtrsm (SIDE, UPLO, TRANSA, DIAG, M, N, ALPHA, A, LDA, B, LDB)
7. Calcular la R de una Descomposición QR: $QR = Z$ LAPACK
SUBROUTINE DGEQRF(M, N, A, LDA, TAU, WORK, LWORK, INFO)
8. Resolver el sistema de ecuaciones triangular superior BLAS2
dtrsv (UPLO, TRANS, DIAG, N, A, LDA, X, INCX)
9. Resolver el sistema de ecuaciones triangular inferior BLAS2
dtrsv (UPLO, TRANS, DIAG, N, A, LDA, X, INCX)

Prestaciones

- The testbed system used in the experimentation is composed by one Intel Xeon E5530 Quad-Core processors (4 cores) running at 2.40 GHz with Ubuntu Linux distro (10.04.2 LTS) as operating system. The high performance **implementation of LAPACK** was provided by **Intel MKL** (*Mathematical Kernel Library, version 10.3*). Finally, experiments reported in this section employ ieee 754 double precision arithmetic.
- OpenMP** is used when possible.

Execution time for the classical and CMB algorithms



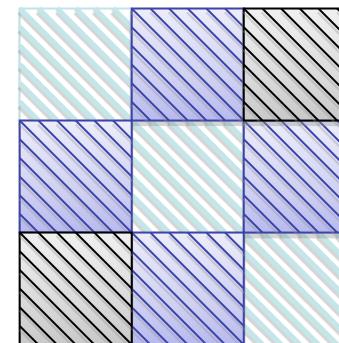
Algorithm/cores	1	2	3	4
Classical	$5.71e + 03$	$3.71e + 03$	$2.67e + 03$	$2.07e + 03$
CMB	$1.54e + 02$	$7.90e + 01$	$5.32e + 01$	$4.09e + 01$

Mejorando la Descomposición de Cholesky: Uso de *StructPack*

- La principal limitación en tiempo y en memoria del Algoritmo anterior es la Descomposición de Cholesky $\xi = L * L^T$, $O(N^3)$, con N grande
- La matriz ξ es una matriz Toeplitz por bloques, y cada bloque es a su vez una matriz Toeplitz.
- Se puede aprovechar su estructura y encontrar esta Descomposición utilizando la librería especializada *StructPack*
- *StructPack* permite calcular la Descomposición de Cholesky **en la mitad de tiempo**: ***Subroutine dtspg_sv from StructPack***
- Las matrices **se almacenan de forma más eficiente** permitiendo resolver problemas de mayor tamaño



Matriz de Toeplitz



Matriz de Toeplitz por bloques
con bloques Toeplitz

Conclusiones

- Las Librerías de Altas Prestaciones pertenecen al ámbito de la Computación Paralela más próximo a las aplicaciones y son la forma más efectiva de abordarlas
- Su uso es totalmente recomendable y sus prestaciones difícilmente superables
 - En particular, en Algebra Lineal Numérica han permitido una técnica de programación orientada al uso de Librerías/Componentes muy útil y eficiente
 - Representan una buena forma de organizar los resultados en desarrollo de software

Referencias

- LAPACK: <http://www.netlib.org/lapack/>
- ScaLAPACK: <http://www.netlib.org/scalapack/>
- BLAS: <http://www.netlib.orgblas/>
- StructPack: <http://www.inco2.upv.es/structpack.html>
- “**Matrix Computations**”. G.Golub & C.Van Loan. 3rd Ed. John Hopkins Univ. Press. 1996
- “**Fast Reliable Algorithms for Matrices with Structure**”. T.Kailath & A.H.Sayed. SIAM 1999