Librerías matriciales paralelas ScaLAPACK

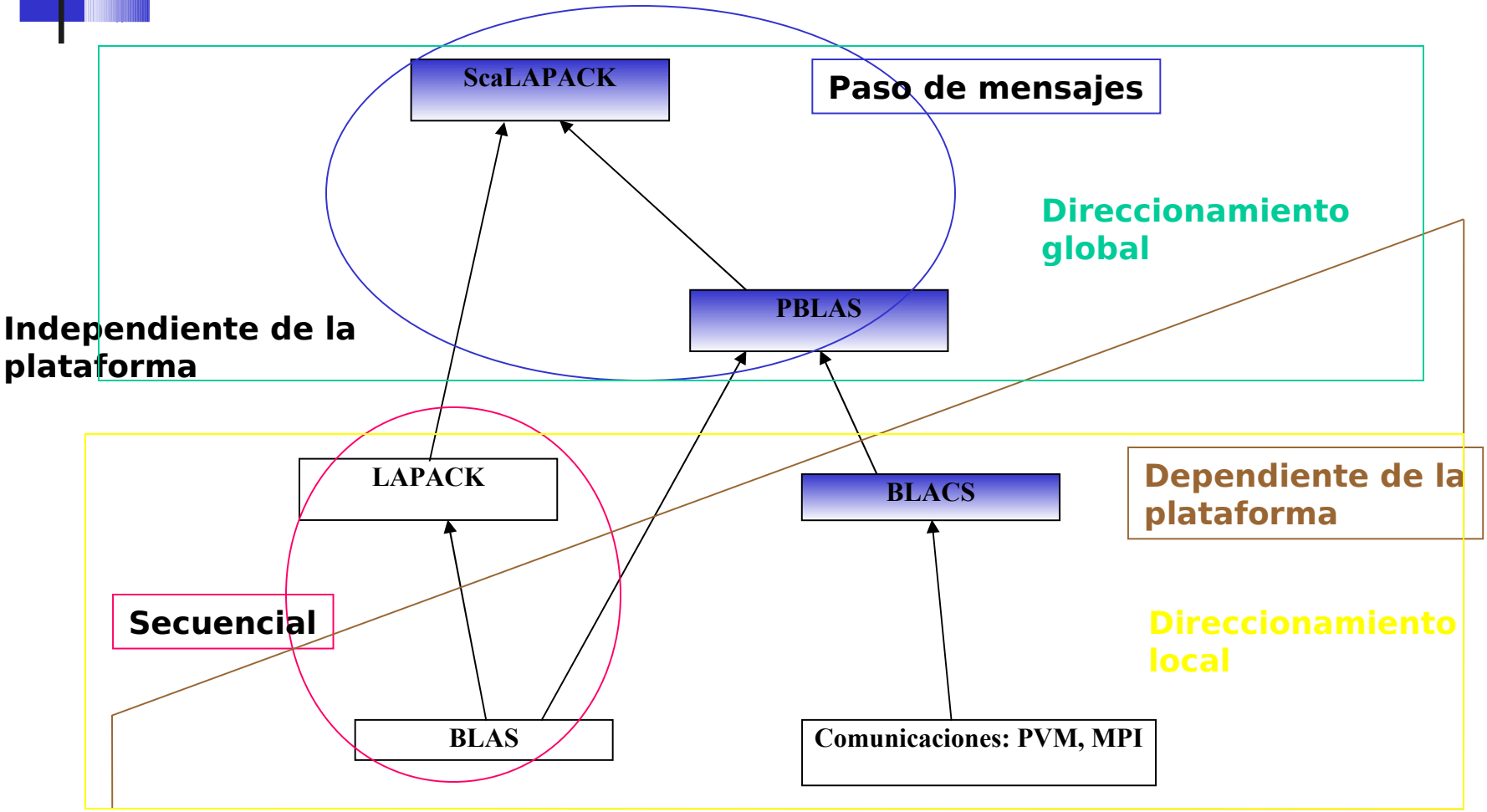
Javier Cuenca
Dpto. de Ingeniería y Tecnología de Computadores
Domingo Giménez
Dpto. de Informática y Sistemas

Facultad de Informática
http://dis.um.es/~domingo
Universidad de Murcia

Contenido

- Introducción
- Librería BLACS
- Librería PBLAS
- Librería ScaLAPACK

Introducción



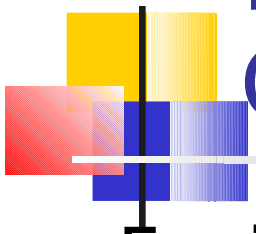
Librería BLACS
(Basic Linear Algebra Communication Subprograms)

- “Basic Linear Algebra Communication Subprograms”
- Hace uso de los paquetes software de paso de mensajes.
- Proporciona facilidad de uso y portabilidad para la comunicación en problemas de álgebra lineal.

Librería BLACS.
Motivación

- Hay varios paquetes que proporcionan una interfaz de paso de mensajes que permanecen invariables a través de varias plataformas (ej. PVM, MPI, PICL).

- Pero, estos paquetes son librerías generales y sus interfaces no son tan fáciles de usar en álgebra lineal como sería deseable.



Librería BLACS.
Objetivos

- Facilidad de programación
  - Simplifica el paso de mensajes y así reduce los errores de programación.
- Facilidad de uso
  - Está a un nivel en el que es fácilmente utilizable por los programadores de álgebra lineal.
- Portabilidad
  - Proporciona una interfaz que está soportada por un gran abanico de computadores paralelos.

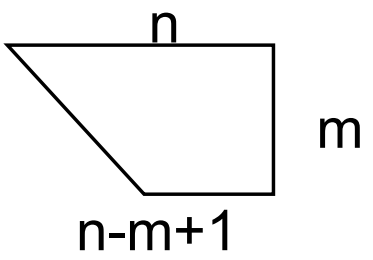
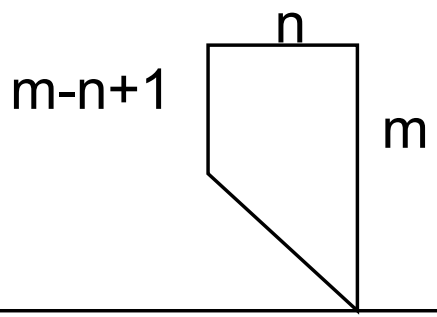
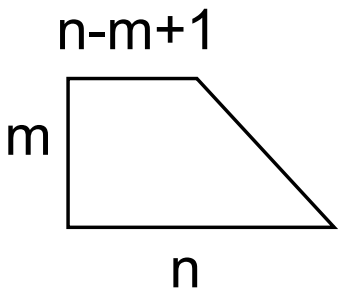
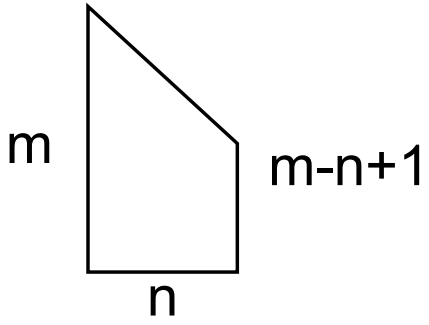
Librería BLACS.
Resultado

- Con BLACS la interfaz y los métodos de uso de las rutinas, han sido especializados y por tanto simplificados.

Librería BLACS.
Ideas base

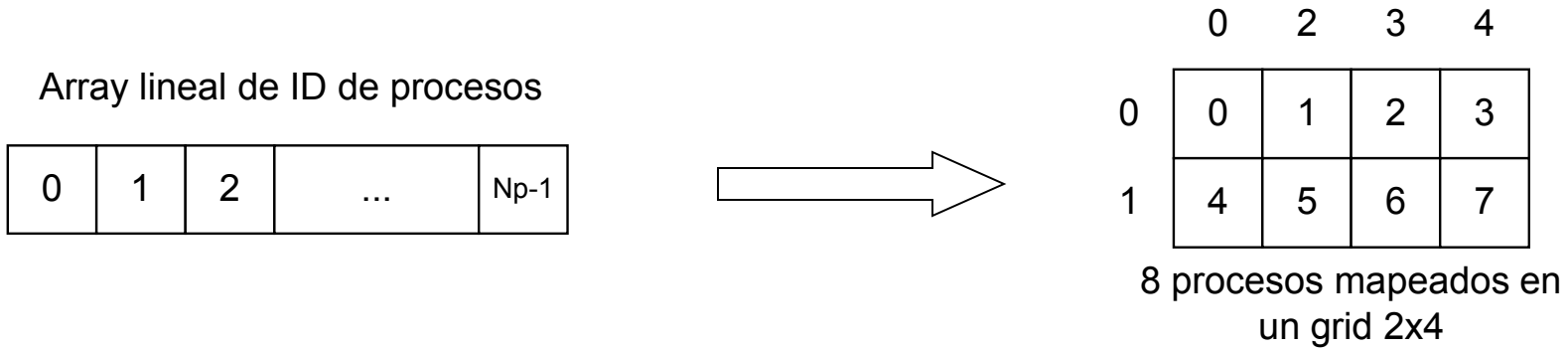
- Comunicación de matrices
  - Generales
    - M filas por N columnas
    - LDA: Distancia entre dos elementos fila consecutivos (Column-major).
  - Trapezoidales, definidas por
    - MxN, LDA
    - UPLO, DIAG (‘U’,’N’)

Librería BLACS.
Ideas base.
Parámetro UPLO

UPLO	M <= N	M>N
'U'		
'L'		

Librería BLACS.
Ideas base

- Mapeo de los procesos en un grid



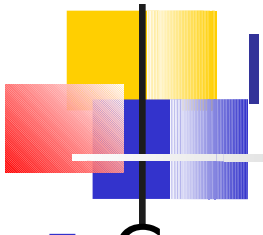
Un conjunto de procesos 0,1,...,Np-1
se mapea como una malla de PxQ procesos,
donde PxQ = Ng <= Np

Librería BLACS.
Ideas base

- Ámbito de una operación en una malla 2D

Ámbito	Significado
Fila	Todos los procesos de la fila participan en la operación
Columna	Todos los procesos de la columna participan en la operación
Todos	Todos los procesos participan en la operación

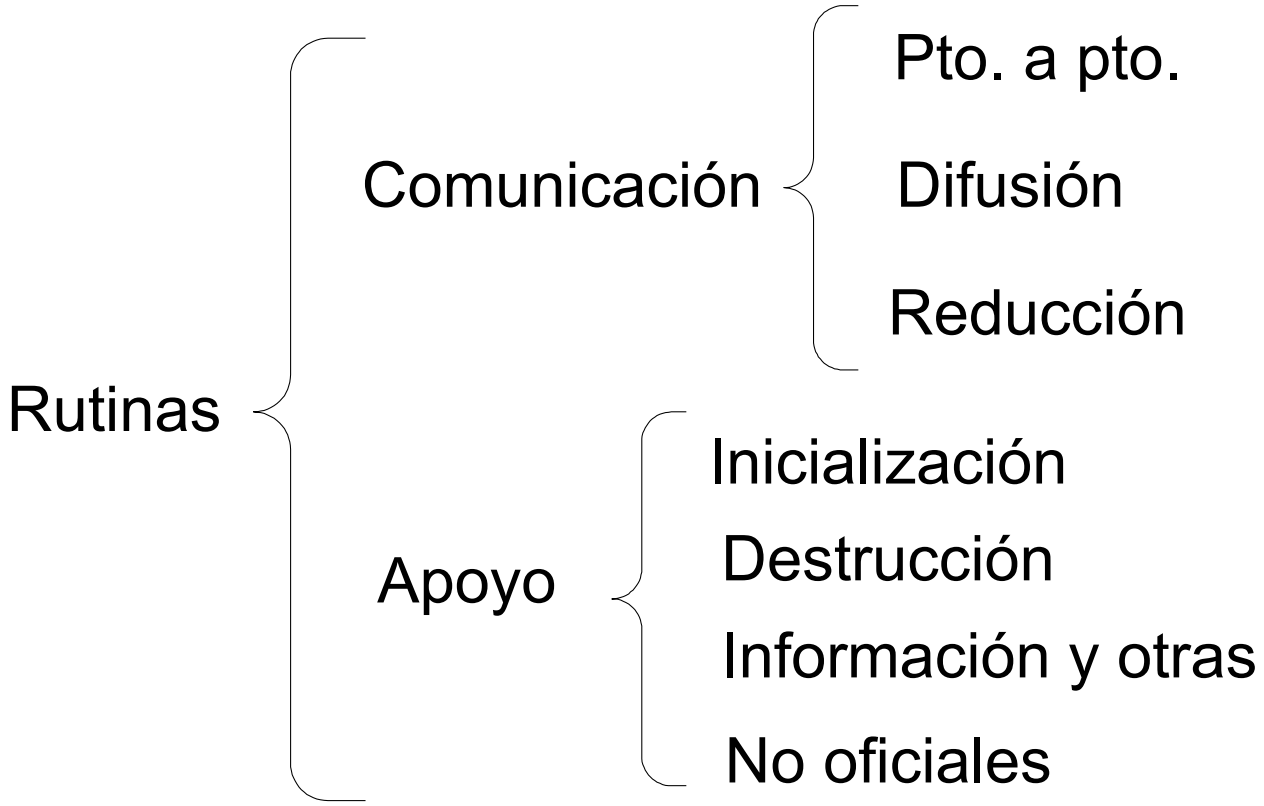
Librería BLACS.
Ideas base



- Contextos
  - Cada malla lógica de procesos está contenida en un contexto BLACS.
  - Los contextos permiten:
    - Crear grupos de procesos arbitrarios
    - Crear grids solapados o disjuntos
    - Aislar los procesos de los grids para que no interfieran con los de otros grids.

Librería BLACS.

Rutinas



Librería BLACS.
Rutinas.
Comunicación Pto a pto

- Nomenclatura de las rutinas punto a punto y difusión

vXXYY2D	
Tipo de datos: v	I (Integer), S, D, C, Z
Forma de la matriz: XX	GE TR
Tipo Comunicación: YY	SD, RV (Send, Recieve) BS, BR (Bcast send, recv)

Librería BLACS.
Rutinas.
Comunicación Pto a pto

- Envío
  - vGESD2D(ICONTXT,M,N,A,LDA,RDEST,CDEST)
  - vTRSD2D(ICONTXT,UPLO,DIAG,M,N,A,LDA,RDEST,CDEST)
- Recepción
  - vGERV2D(ICONTX,M,N,A,LDA,RSRC,CSRC)
  - vTRRV2D(ICONTX,UPLO,DIAG,M,N,A,LDA,RSRC,CSRC)

Librería BLACS.
Rutinas.
Difusión

- Envío
  - vXXBS2D(ICONTX,SCOPE,TOP[,UPLO,DIAG],M,N,A, LDA)
- Recepción
  - vXXBR2D(ICONTX,SCOPE,TOP[,UPLO,DIAG],M,N,A, LDA,RSRC,CSRC)

Librería BLACS. Rutinas. Difusión - Topologías

- Basadas en anillo (pipelining)
  - Anillo unidireccional
  - Anillo dividido
  - Multianillo
- Basadas en árboles (non-pipelining)
  - Hipercubo
  - Árbol general
- General Tree Gather
- Intercambio bidireccional
- http://www.netlib.org/blacs/BLACS/Top.html#BSRing

Librería BLACS.
Rutinas.
Difusión

- Máximo
  - vGAMX2D(ICONTX,SCOPE,TOP,M,N,A,LDA,RA,CA,RCFLAG,RDEST,CDEST)
- Mínimo
  - vGAMN2D(ICONTXT,SCOPE,TOP,M,N,A,LDA,RA,CA,RCFLAG,RDEST,CDEST)
- Suma
  - vGSUM2D(ICONTXT,SCOPE,TOP,M,N,A,LDA,RDEST,CDEST)

Librería BLACS. Rutinas. Apoyo

- Inicialización
  - BLACS_PINFO: usada cuando se necesita alguna información inicial del sistema. Devuelve el índice de proceso y el número de procesos.
  - BLACS_SETUP: esta solo tiene sentido en PVM que permite añadir dinámicamente procesos.
  - BLACS_GRIDINIT: mapea los procesos disponibles en el grid BLACS. Devuelve el contexto.
  - BLACS_GRIDMAP: establece como se mapearán los procesos coordenados BLACS respecto de la numeración nativa de la máquina.

Librería BLACS.
Rutinas.
Apoyo

- Destrucción
  - BLACS_FREEBUF: libera los buffers usados por BLACS.
  - BLACS_GRIDEXIT: libera un contexto BLACS.
  - BLACS_ABORT: utilizada para abortar un proceso, normalmente se llama cuando ocurre un error.
  - BLACS_EXIT: esta rutina se debe llamar cuando un proceso ha terminado de usar BLACS. Se puede indicar si la capa inferior de comunicación se va a seguir usando o si se puede liberar (pvm_exit).

Librería BLACS.
Rutinas.
Apoyo

- Información y otras:
  - BLACS_GRIDINFO: devuelve información acerca de un proceso en un determinado contexto.
  - BLACS_PNUM: devuelve el número de proceso del sistema dado su posición en el grid.
  - BLACS_PCOORD: dado el número de proceso del sistema devuelve sus coordenadas en el grid.

Librería BLACS.

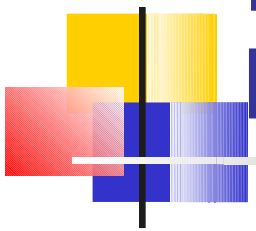
Rutinas.

Apoyo

- BLACS_GET: devuelve información interna de BLACS (context handle, rango de IDs de mensajes, nivel de depuración,...).
- BLACS_SET: establece valores internos de parámetros BLACS.

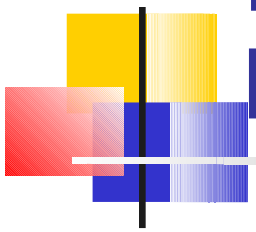
Librería BLACS.
Rutinas.
Apoyo

- Rutinas no oficiales
  - No forman parte del BLACS estándar.
  - Pueden no estar en todas las implementaciones de BLACS.
  - Ejemplos:
    - SETPVMTIDS
    - DCPUTIME
    - DWALLTIME



Librería BLACS.
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



Librería BLACS.
Ejemplos

- Cscala_01.c
- Cscala_02.c

Librería BLACS.
Ejemplo Cscala_01.c

- `blacs_pinfo_(&iam,&nprocs);`
  - &iam (o): quién soy yo
  - &nprocs (o): total de procesos disponibles, antes de inicializar BLACS

- `blacs_get_(&i_cero,&i_cero,&contxt);`
  - &i_cero (i): si estoy en un contexto ya, si no, se ignora
  - &i_cero (i): qué devolver en 3er argumento. =0 un handler del contexto de sistema por defecto
  - &contxt (o): El handler del contexto sistema por defecto

  - *Obtiene el handler del contexto del sistema*

Librería BLACS.
Ejemplo Cscala_01.c

- `blacs_gridinit_(&contxt,"R",&nprow,&npcol);`
  - &contxt (i/o):
    - input: el handler que indica el contexto del sistema (blacs_get)
    - Output: el nuevo handler para el contexto de BLACS
  - “R” (i): row-major en el número de procesos en el grid
  - &nprow (i): cuántas filas de procesos en el grid
  - &npcol (i): cuántas columnas de procesos en el grid
  - *Inicializa el grid de procesos*

Librería BLACS.
Ejemplo Cscala_01.c

- `blacs_gridinfo_(&contxt,&nprow,&npcol,&myprow,&mypcol);`
  - &contxt (i): el handler del contexto
  - &nprow (o): número de filas de procesos en el grid
  - &npcol (o): número de columnas de procesos en el grid
  - &myprow (o): mi fila en el grid
  - &mypcol (o): mi columna en el grid

  - *Obtiene informacion del grid existente*

Librería BLACS.
Ejemplo Cscala_01.c

- `icaller=blacs_pnum_(&contxt,&myprow,&mypcol);`
  - &contxt (i): handler del contexto
  - &myprow (i): mi número de fila
  - &mypcol (i): mi número de columna

  - *Devuelve el número de proceso del sistema correspondiente a las coordenadas que se le pasa*

Librería BLACS.
Ejemplo Cscala_01.c

- `igerv2d_(&contxt,&i_uno,&i_uno,&icaller,&i_uno,&i,&j)`
  - &contxt (i):
  - &i_uno (i): numero filas matriz a recibir
  - &i_uno (i): numero columnas matriz a recibir
  - &i_caller (o): matriz a recibir
  - &i_uno (i): lead dimension
  - &i (i): nº fila del proceso fuente del envio
  - &j (i): nº columna del proceso fuente del envio

  - *Recibe una matriz de datos*

Librería BLACS.
Ejemplo Cscala_01.c

- `blacs_pcoord_(&contxt,&icaller,&hisrow,&hiscol);`
  - &contxt
  - &icaller (i): número de proceso
  - &hisrow (o): número de fila del proceso
  - &hiscol (o): número de columna del proceso

  - *Dado un número de proceso devuelve sus coordenadas en el grid*

Librería BLACS.
Ejemplo Cscala_01.c

- `blacs_abort_(&contxt,&i_uno);`
  - &contxt(i)
  - &i_uno (i): número de error por el que se detiene

  - *Finaliza un proceso BLACS*

Librería BLACS.
Ejemplo Cscala_01.c

- `igesd2d_(&contxt,&i_uno,&i_uno,&icaller,&i_uno,&i_cero,&i_cero);`
  - &contxt (i)
  - &i_uno: número de filas de la matriz
  - &i_uno: número de columnas de la matriz
  - &i_caller: la matriz que se envía
  - &i_uno: lead dimension de la matriz
  - &i_cero: coordenada fila de proceso destino
  - &i_cero: coordenada columna de proceso destino

  - *Envía una matriz de datos a otro proceso*

Librería BLACS. Ejemplo Cscala_01.c

- Compilación

```
$make Cscala_01

mpicc -O4  Cscala_01.c -o Cscala_01
  /usr/lib64/libmpiblacsCinit.a
  /usr/lib64/libmpiblacsF77init.a
  /usr/lib64/libmpiblacs.a -lmpi -llapack -lblas -lm
```

- Ejecución

```
mpirun –np num_procesos Cscala_01 num_fil num_col
```

Librería BLACS. Ejemplo Cscala_01.c

- Comp

```
$make Csc
mpicc -O4
  /usr/li
  /usr/li
  /usr/li
```

- Ejecu

```
mpirun -n
```

```
[javiercm@sol ejemplos]$ mpirun -np 4 Cscala_01 2 2

>>>>BLACS abierto: 4 procesos<<<<<<

GRID INICIADA: proceso 1, fila 0, columna 1

GRID INICIADA: proceso 2, fila 1, columna 0

GRID INICIADA: proceso 3, fila 1, columna 1

GRID INICIADA: proceso 0, fila 0, columna 0

yo (0,0), he recibido de proceso (0,1) el mensaje [1]

yo (0,0), he recibido de proceso (1,0) el mensaje [2]

yo (0,0), he recibido de proceso (1,1) el mensaje [3]

ESTO JUNCIONA!!: Chequeados todos los procesos

yo, proceso (0,1), voy a enviar al proceso (0,0) mi identificadoryo, proceso
(1,0), voy a enviar al proceso (0,0) mi identificadoryo, proceso (1,1), voy
a enviar al proceso (0,0) mi identificador

>>>>BLACS cerrado<<<<<<
```

Librería BLACS.
Ejemplo Cscala_02.c

- `dgebs2d_(&contxt,p_all,p_vacio,&N,&N,A,&ldA);`
  - &contxt
  - P_all (i): scope del broadcast: “All”, “Row”, “Column”
  - P_vacio (i): topología del algoritmo de broadcast, “ “ es la de por defecto más eficiente.
  - &N,&N,A,&ldA: la matriz y sus dimensiones

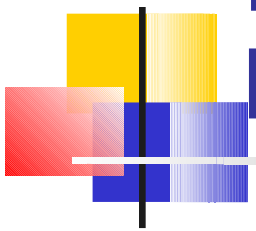
  - *Broadcast de una matriz, el emisor*

Librería BLACS.
Ejemplo Cscala_02.c

- dgebr2d_
  (&contxt,p_all,p_vacio,&N,&N,A,&ldA,&i_Cero,
  &i_Cero);
  - &contxt
  - P_all (i): scope del broadcast: “All”, “Row”, “Column”
  - P_vacio (i): topología del algoritmo de broadcast, “ “ es la de por defecto más eficiente.
  - &N,&N,A,&ldA: la matriz y sus dimensiones
  - &i_Cero,&i_Cero (i): coordenadas del proceso que hace el envío

  - *Broadcast de una matriz, el receptor*

Librería BLACS.
Ejemplo Cscala_02.c



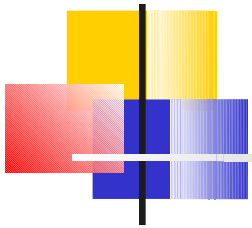
- `Blacs_barrier_(&contxt,p_all);`
  - &contxt
  - P_all (i): scope del broadcast: “All”, “Row”, “Column”
  - *Barrera para que todo el mundo espere hasta que llegue el proceso más lento*

Librería BLACS.
Ejemplo Cscala_02.c

- `dgsum2d_(&contxt,p_all,p_vacio,&N,&N,A,&ldA,&i_Cero,&i_Cero);`
  - &contxt
  - P_all (i): scope del broadcast: “All”, “Row”, “Column”
  - P_vacio (i): topología del algoritmo de broadcast, “ “ es la de por defecto más eficiente.
  - &N,&N,A,&ldA: la matriz y sus dimensiones
  - &i_Cero,&i_Cero (i): coordenadas del proceso que hace el envío

- *Acumulación sumando de una matriz. Lo que ejecutan todos, en el proceso receptor (el [0,0] en este caso) se acumulan los datos sobre su copia de la matriz A*

Librería BLACS. Ejemplo Cscala_02.c



- Compilación

```
$make Cscala_02

mpicc -O4  Cscala_02.c -o Cscala_02
  /usr/lib64/libmpiblacsCinit.a
  /usr/lib64/libmpiblacsF77init.a
  /usr/lib64/libmpiblacs.a -lmpi -llapack -lblas -lm
```

- Ejecución

mpirun –np *num_procesos* Cscala_02 *num_fil num_col*

Librería BLACS. Ejemplo Cscala_02.c

- Comp

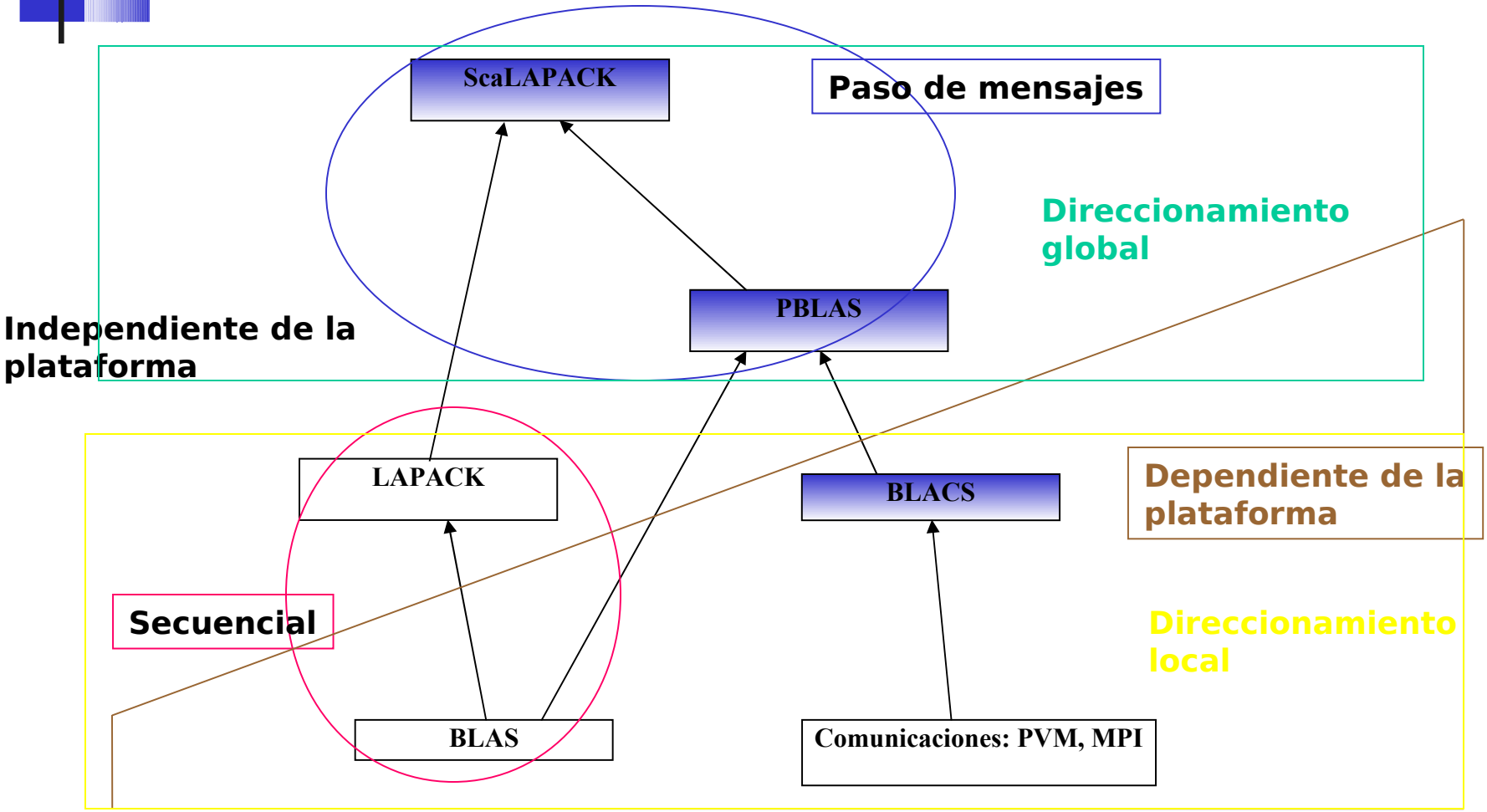
```
$make Csc
mpicc -O4
  /usr/li
  /usr/li
  /usr/li
```

- Ejecu

```
[javiercm@sol ejemplos]$ mpirun -np 4 Cscala_02 2 2

>>>>BLACS abierto: 4 procesos<<<<<<

Procesador 0, matriz recibida tras acumulacion:
6.00    6.00    6.00    6.00
6.00    6.00    6.00    6.00
6.00    6.00    6.00    6.00
6.00    6.00    6.00    6.00

>>>>BLACS cerrado<<<<<<
```

mpirun –np *num_procesos* Cscala_02 *num_fil num_col*

Librería PBLAS
(Parallel Basic Linear Algebra Subprograms)



Librería PBLAS
(Parallel Basic Linear Algebra Subprograms)

- PBLAS ofrece a ScaLAPACK rutinas equivalentes a las ofrecidas por BLAS a LAPACK.

- Es un conjunto de rutinas para memoria distribuida análogas a las de BLAS.

Librería PBLAS
(Parallel Basic Linear Algebra Subprograms)

- Se consigue que el código de ScaLAPACK sea bastante parecido al de LAPACK.

- Objetivo: que PBLAS proporcione un estándar para memoria distribuida, como lo hace BLAS para memoria compartida.

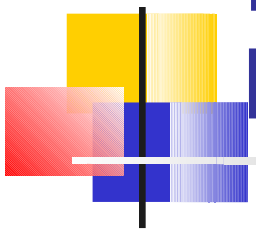
Librería PBLAS.
Niveles

- Los niveles de PBLAS son los mismos que los de BLAS:
  - Nivel 1: Operaciones vector-vector. Coste O(n)
  - Nivel 2: Operaciones matriz-vector. Coste $O(n^2)$
  - Nivel 3: Operaciones matriz-matriz. Coste $O(n^3)$

Librería PBLAS.
Nomenclatura

- El nombre de las rutinas es el mismo que en BLAS anteponiendo la letra P.

PXYYZZZ (PBLAS 2 y 3)	
Tipo de datos: X	S, D, C, Z
Tipo de matriz: YY	GE,SY,HE,TR
Operación: ZZZ	MM, MV, R, R2, RK, R2K, SM, SV



Librería PBLAS.
Disposición de los datos

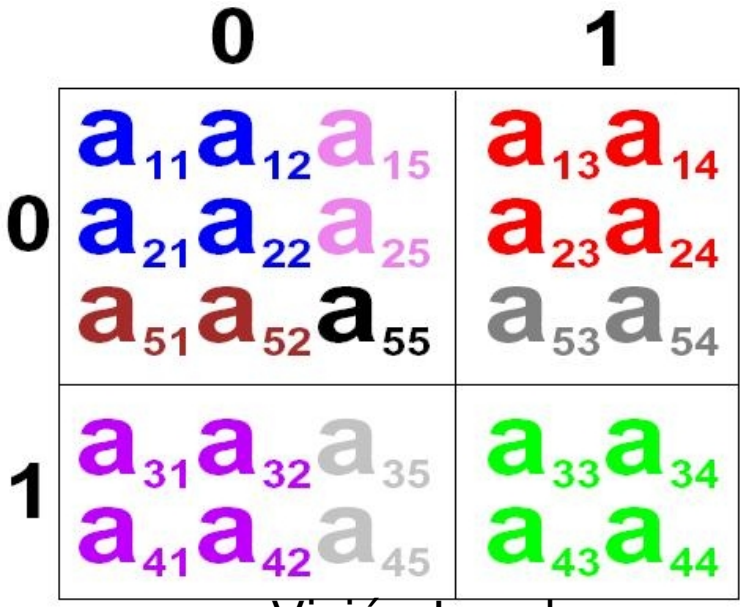
- Distribución de las matrices
  - Descomposición cíclica por bloques
  - Descriptores
    - Para describir las matrices distribuidas se requieren algunos datos, estos se encuentran englobados en los descriptores (array descriptor `DESC_`).

Librería PBLAS.

Disposición de los datos.

Descomposición cíclica por bloques

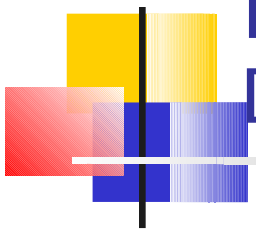
Visión Global
Matriz 5x5 particionada
en bloques de 2x2.



Visión Local
Grid de 2x2 procesos.
Datos locales.

Garantiza un buen balance de la carga → Rendimiento y escalabilidad

Librería PBLAS.
Disposición de los datos.
Descriptores



- El descriptor de arrays encapsula la información necesaria para describir una matriz distribuida.

- El descriptor se puede inicializar llamando a la rutina `descinit`

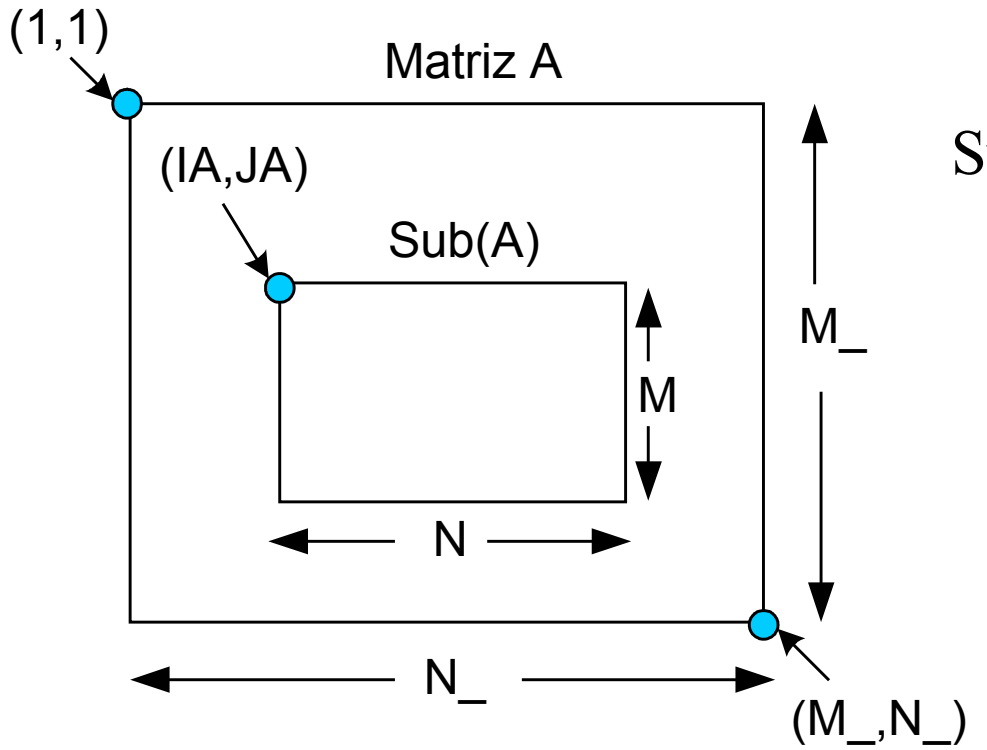
Librería PBLAS.

Disposición de los datos.

Descriptores

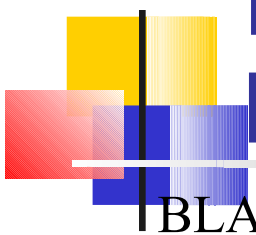
Campo	Significado
TYPE	El tipo de descriptor
CTXT	Contexto
M	Filas globales
N	Columnas globales
MB	Tamaño del bloque de filas
NB	Tamaño del bloque de columnas
RSRC	Fila del proceso con la primera fila
CSRC	Columna del proceso con la primera columna
LLD	Dimensión principal

Librería PBLAS.
Visión global de las matrices

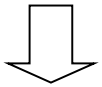


Sub(A) =
A(IA:IA+M-1;
JA:JA+N-1)

Librería PBLAS.
De BLAS a PBLAS

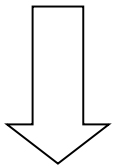


BLAS → DGEXXX(M,N,A(IA,JA),LDA,...)

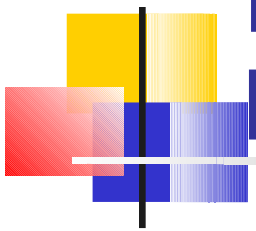


PBLAS → PDGEXXX(M,N,A,IA,JA,DESCA,...)

BLAS → DGEMM(TRANSA,TRANSB,M,N,K,ALPHA
A,LDA,
B,LDB, BETA,
C,LDC)



PBLAS → PDGEMM(TRANSA,TRANSB,M,N,K,ALPHA
A,IA,JA,DESCA,
B,IB,JB,DESCB, BETA,
C,IC,JC,DESCC)



Librería PBLAS.
Referencias

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

Librería PBLAS.
Ejemplo: Cscala_03.c

- descinit

```
_(DESCA,&N,&N,&R,&R,&i_cero,&i_cero,&contxt,&ldlA,&info
);
```

  - *Crea el descriptor de una matriz que se va a distribuir (ScaLAPACK)*
  - DESCA (o), el descriptor creado
  - &N, &N (i): número de filas y columnas de matriz global
  - &R, &R (i): número de filas y columnas de bloque de distribución
  - &i_cero, &i_cero (i): coordenadas en el grid del proceso sobre el que distribuir primera fila/columna de la matriz
  - &contxt
  - &ldaA (i): leading dimension de la matriz local donde guardar bloques
  - &info (o): =0 éxito

Librería PBLAS.
Ejemplo: Cscala_03.c

- pdgemm

```
_(p_normal,p_normal,&N,&N,&N,&alpha,lA,&ia,&ja,DESCA,lB
,&ia,&ja,DESCB,&beta,lC,&ia,&ja,DESCC)
```

  - P_normal, p_normal: no traspuesta
  - &N, &N, &N: dimensiones de las matrices
  - &alpha: escalar que multiplica a A·B
  - lA: parte local de la matriz A
  - &ia, &ja: primera fila y columna global a operar
  - DESCA: descriptor de la matriz A
  - lB: parte local de la matriz B
  - &ia, &ja: primera fila y columna global a operar
  - DESCB: descriptor de la matriz B
  - &beta: escala que multiplica a matriz C
  - lC: parte local de la matriz C
  - &ia, &ja: primera fila y columna global a operar
  - DESCC: descriptor de la matriz C

Librería PBLAS.
Ejemplo: Cscala_03.c

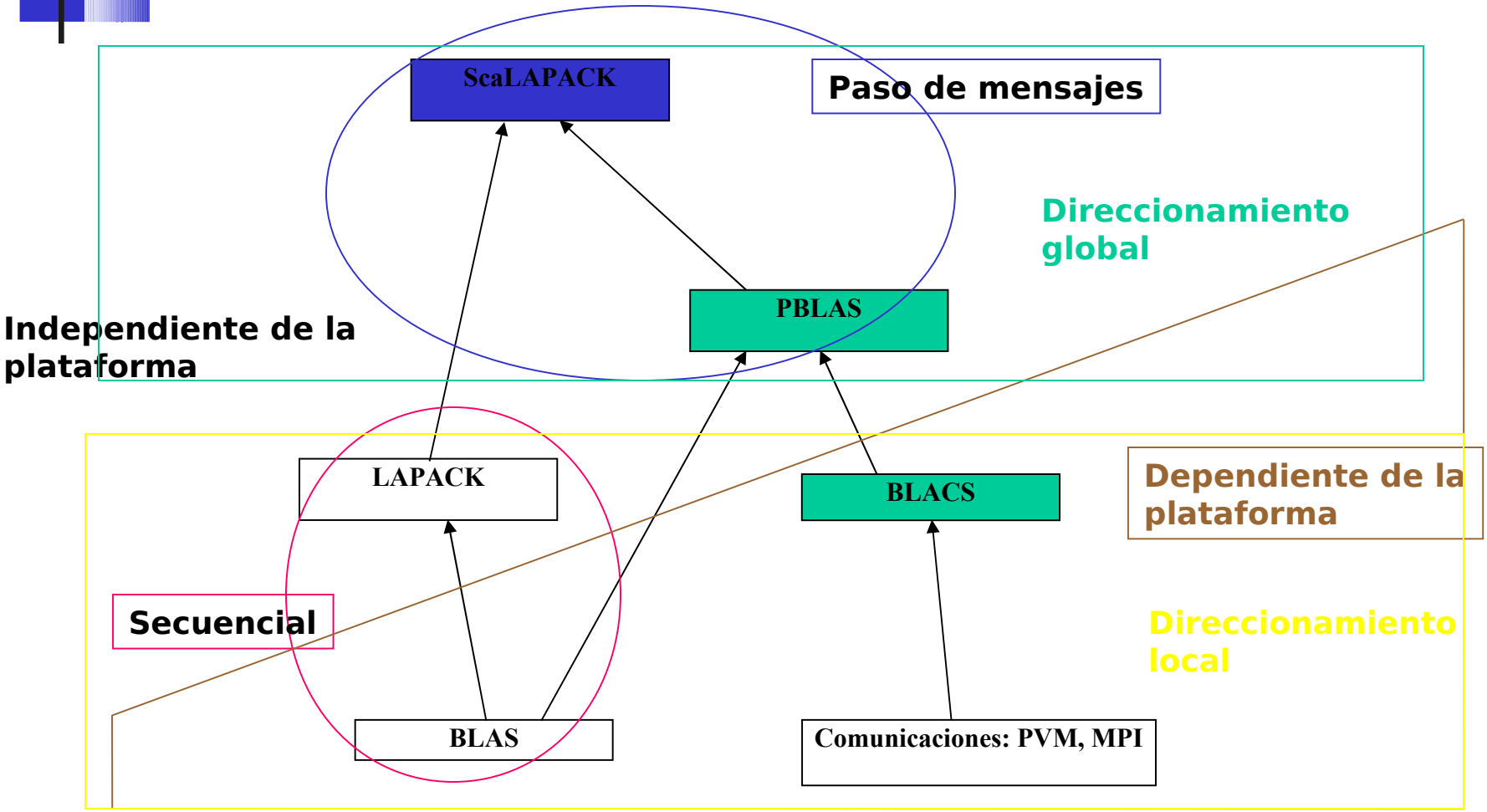
- Compilación

```
$ make scala_03
mpicc -O4  Cscala_03.c gm.o -o Cscala_03
-lscalapack
/usr/lib64/libmpiblacsCinit.a
/usr/lib64/libmpiblacsF77init.a
/usr/lib64/libmpiblacs.a -lmpi -llapack -lblas -lm
```

Ejecución

```
mpirun –np num_procesos Cscala_03 num_fil num_col
```

Librería ScaLAPACK (Scalable LAPACK)



Librería ScaLAPACK (Scalable LAPACK)

- “Scalable Linear Algebra PACKage”

- Scalable LAPACK, es una librería software para la computación paralela de álgebra lineal en computadores con memoria distribuida.

- Esta librería incluye un subconjunto de rutinas LAPACK rediseñadas para computadores paralelos MIMD con memoria distribuida.

Librería ScaLAPACK (Scalable LAPACK)

- Está escrito al estilo Single-Program-Multiple-Data.
- Usa el paso de mensajes explícito para la comunicación entre procesos.
- ScaLAPACK asume que las matrices están distribuidas de forma **cíclica por bloques**.
  - Eficiencia
  - Escalabilidad
  - Distribución homogénea de la carga

Librería ScaLAPACK (Scalable LAPACK)

- Las rutinas se basan en algoritmos que trabajan por bloques
  - (minimizar los movimientos de datos).
- Bases de construcción de ScaLAPACK:
  - La versión distribuida de BLAS, PBLAS.
  - BLACS para las tareas de comunicación.

Librería ScaLAPACK.
Objetivos

- Eficiencia (BLAS 3)
- Fiabilidad (LAPACK, calculo de cotas de error)
- Escalabilidad (Nuevos algoritmos frente a los que no escalaban de LAPACK)
- Portabilidad (Dependencia de la máquina: BLAS, BLACS)
- Flexibilidad (Herramientas para el álgebra)
- Facilidad de uso (Similar a LAPACK)

Librería ScaLAPACK.
Eficiencia y portabilidad

- ScaLAPACK puede ejecutarse en cualquier máquina que soporte PVM o MPI.

- La eficiencia de ScaLAPACK depende de implementaciones eficientes de BLAS y BLACS (Se deben usar librerías BLAS y BLACS optimizadas para cada máquina).

Librería ScaLAPACK.
Estructura

- La estructura es análoga a la de LAPACK.

- Rutinas conductoras: son las que resuelven un problema completo.
  - Ejemplos:
    - Sistema lineal de ecuaciones.
    - Valores propios de una matriz real simétrica.

Librería ScaLAPACK.
Estructura

- Rutinas computacionales: realizan tareas computacionales dispares.
  - Pueden ser usadas para realizar tareas que no pueden ser abordadas convenientemente con las rutinas conductoras.
  - Ejemplos:
    - Descomposición LU.
    - Reducción de una matriz real simétrica a la forma tridiagonal.

Librería ScaLAPACK. Estructura

- Rutinas auxiliares. Se dividen en:
  - Rutinas que realizan subtareas de algoritmos por bloques.
  - Rutinas que realizan cálculos de bajo nivel usados habitualmente.
    - Ej. Escalado de una matriz, cálculo de la norma matricial, etc.
  - Algunas extensiones de PBLAS.

Librería ScaLAPACK.
Nomenclatura

PXYYZZZ (R. conductoras y computacionales)	
Tipo de datos: X	S : REAL D : DOUBLE PRECISION C : COMPLEX Z : DOUBLE COMPLEX
Tipo de matriz: YY	GE,DB,GB,DT,PO,PB,PT,HE,SY.
Operación: ZZZ	SV,EV,SVD,GVX,TRF,CON,...

Librería ScaLAPACK.
Conclusiones

- ScaLAPACK no resuelve algunos problemas que se resolvían con LAPACK.
  - Problema no simétrico de valores propios (NEP).
  - Problema generalizado no simétrico de valores propios (GNEP).
  - Problema generalizado de la descomposición en valores singulares (GSVD).

Librería ScaLAPACK.
Conclusiones

- Sucede a LAPACK diseñado para memoria distribuida.

- Se han logrado los objetivos mediante BLAS y PBLAS.

Librería ScaLAPACK.
Trabajo futuro

- Algunos trabajos futuros de ScaLAPACK:
  - Out-of-core eigensolvers.
  - Rutinas divide y vencerás.
  - Interfaces C++ y Java.
  - Incrementar la flexibilidad y usabilidad.

Librería ScaLAPACK.
Referencias

- Página principal: http://www.netlib.org/scalapack/scalapack_home.html

- User's guide v1.7: http://www.netlib.org/scalapack/slug/index.html

Librería ScaLAPACK.
Ejemplo: Cscala_qr.c

- pdgeqrf_

```
(&N,&N,lA,&ia,&ja,DESCA,taus,work,&lwork,&info);
```

  - &N, &N: dimensionesiones de la matriz A
  - lA: parte local de la matriz A
  - &ia, &ja: primera fila y columna global de la matriz a operar
  - DESCA: descriptor de la matriz A
  - Taus: los factores escalares de los reflectores elementares
  - Work: espacio de trabajo local
  - &lwork: dimension de work
  - Info: si ha habido errores o no

Librería ScaLAPACK.
Ejemplo: Cscala_qr.c

- Compilación

```
$ make Cscala_qr
mpicc -O4  Cscala_qr.c dis_con.o gm.o -o Cscala_qr
  -lscalapack  /usr/lib64/libmpiblacsCinit.a
  /usr/lib64/libmpiblacsF77init.a
  /usr/lib64/libmpiblacs.a -lmpi -llapack -lblas -lm
```

- Ejecución

mpirun –np *num_procesos* Cscala_qr *num_fil num_col tama_ini tama_fin inc_tama tamblo_ini tamblo_fin inc_tamblo*

Librería ScaLAPACK.
Ejemplo: Cscala_qr.c

- Compilación

```
$ make Csc
mpicc -O4                                              r
  -lscalap
  /usr/lib
  /usr/lib                                             -lm
```

- Ejecuc

```
mpirun -np n                                           ini
tama_fin inc
```

```
$ mpirun -np 4 Cscala_qr 2 2 100 400 2 25 50 2
>>>>BLACS abierto: 4 procesos<<<<<<


np=4 NFN=2 NCN=2 N=100 R=25 tiempo=0.648420
np=4 NFN=2 NCN=2 N=100 R=50 tiempo=0.417958
np=4 NFN=2 NCN=2 N=200 R=25 tiempo=1.030247
np=4 NFN=2 NCN=2 N=200 R=50 tiempo=1.049778
np=4 NFN=2 NCN=2 N=400 R=25 tiempo=2.013620
np=4 NFN=2 NCN=2 N=400 R=50 tiempo=2.441342


>>>>BLACS cerrado<<<<<<
```

Librería ScaLAPACK.
Ejemplo: Cscala_lu.c

- `pdgetrf_(&N,&N,lA,&ia,&ja,DESCA,ipiv,&info);`
  - &N, &N: dimensionesiones de la matriz A
  - lA: parte local de la matriz A
  - &ia, &ja: primera fila y columna global de la matriz a operar
  - DESCA: descriptor de la matriz A
  - Ipiv: pivotaje realizado durante la factorización
  - Info: si ha habido errores o no

Librería ScaLAPACK.
Ejemplo: Cscala_lu.c

- Compilación

```
$ make Cscala_lu
mpicc -O4  Cscala_lu.c dis_con.o gm.o -o Cscala_lu
  -lscalapack  /usr/lib64/libmpiblacsCinit.a
  /usr/lib64/libmpiblacsF77init.a
  /usr/lib64/libmpiblacs.a -lmpi -llapack -lblas -lm
```

- Ejecución

dmpirun –np *num_procesos* scala_lu *num_fil num_col tama_ini tama_fin inc_tama tamblo_ini tamblo_fin inc_tamblo*

Librería ScaLAPACK.
Ejemplo: Cscala_lu.c

- Compilación

```
$ make Csca
mpicc -O4
  -lscalap
  /usr/lib
  /usr/lib
```

```
u
-lm
```

- Ejecuci

```
dmpirun -np
tama_ini ta
inc_tamblo
```

```
$ mpirun -np 4 scala_lu 2 2 100 400 2 25 50 2
>>>>BLACS abierto: 4 procesos<<<<<<

np=4 NFN=2 NCN=2 N=100 R=25 tiempo=0.320304
np=4 NFN=2 NCN=2 N=100 R=50 tiempo=0.374014
np=4 NFN=2 NCN=2 N=200 R=25 tiempo=1.089815
np=4 NFN=2 NCN=2 N=200 R=50 tiempo=1.043918
np=4 NFN=2 NCN=2 N=400 R=25 tiempo=1.983347
np=4 NFN=2 NCN=2 N=400 R=50 tiempo=1.912060

>>>>BLACS cerrado<<<<<<
```