Lezione 6

Dalla precedente lezione

- Algoritmi paralleli per le operazioni di base dell'algebra lineare (prod. Matrice-matrice)
- Distribuzione ciclica a blocchi 2D delle matrici
  - bilanciamento del carico e buon rapporto comm/calc
- Obiettivo: sviluppare algoritmi e software per l'algebra lineare parallela
  - Algoritmi di fattorizzazione, autovalori, minimi quadrati

parallelizzare LAPACK

ScaLAPACK (1995)

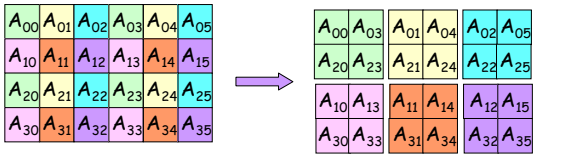
- Rappresenta lo stato dell'arte per quanto riguarda le librerie parallele per l'algebra lineare densa
- Ambienti paralleli a memoria distribuita
- Modello di programmazione SPMD
- Ultima versione 2007
- Algoritmi di
  - Fattorizzazione LU, LLT, QR, …
  - Risoluzione sistemi di equazioni
  - Calcolo di autovalori, minimi quadrati, SVD
- www.netlib.org/scalapack

Obiettivi di ScaLAPACK

- Scalabilita' e prestazioni
  - Moduli locali ottimizzati relativamente al rapporto comunicazioni/calcolo (BLAS)
  - Bilanciamento del carico (distribuzione ciclica a blocchi 2D)
- Portabilita' e modularita'
  - Strumenti consolidati: BLAS, PBLAS, BLACS, LAPACK
- Semplicita' di utilizzo
  - Interfaccia quanto piu' simile possibile a LAPACK

Organizzazione dei dati in ScaLAPACK

Distribuzione ciclica a blocchi 2D



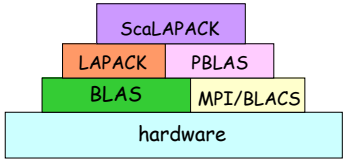
Matrice A

Suddivisione di A su una gliglia 2x3 di processori

- assicura un buon bilanciamento del carico
- Diminuisce le comunicazioni
- buon rapporto comm/calc
- Poco intuitivo

Come parallelizzare LAPACK

- ScaLAPACK si ottiene da LAPACK sostituendo le chiamate di BLAS con opportune chiamate a PBLAS, una libreria parallela per le operazioni di base dell'algebra lineare
- Necessita' di strumenti per implementare facilmente gli algoritmi per le operazioni di base dell' algebra lineare (es. SUMMA per il prodotto matrice matrice)



BLACS

- Basic Linear Algebra Communication Subprograms
- Una libreria di comunicazione sviluppata per diverse piattaforme (MPI, PVM, NX, …)
- Uno strumento per facilitare lo sviluppo di codici paralleli a memoria distribuita per l'algebra lineare
- Orientato alla comunicazione di oggetti 2D (blocchi di matrici)
  - Miglioramento della leggibilita'
- implementazioni ottimizzate per varie piattaforme
  - Miglioramento della portabilita' e dell'efficienza
- Interfaccia Fortran
- http://www.netlib.org/blacs

BLACS

- I processi sono organizzati secondo una griglia 2D per facilitare il mapping dei blocchi di una matrice ai processi
  - Processi identificati da una coppia di indici (myrow,mycol)
- Per facilitare la distribuzione dei blocchi, le operazioni di comunicazione prevedono tre "ambiti" o "scope", in cui partecipano tutti i processi dell'ambirto stesso
- Per una griglia 2D ci sono 3 ambiti naturali
  - Riga, colonna, tutti

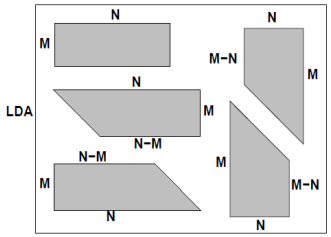
Es. Griglia di Processi 2x3

	0	1	2	3 (colonna)
0	0	1	2	3
1	4	5	6	7
2	8	9	10	11

(riga)

BLACS

- Data una matrice A e' possibile comunicare vari tipi di sottomatrici utili per le operazioni di fattorizzazione



Matrice A

- 3 tipi di routine:
  - Point-to-point, collettive (di comunicazione e calcolo), di supporto

BLACS supporto

- blacs_gridinit(icontxt, order, nprow, npcol)
  - Inizializza una griglia di processori nprow x npcol.
  - order specifica come devono essere mappati i processi (Row major o Column major)
  - icontxt e' il contesto che deve essere usato nelle successive chiamate
- blacs_gridinfo(icontxt, nprow, npcol, myprow, mypcol)
  - Restituisce la dimensione della griglia associata a icontxt
  - Restituisce le coordinate del processo chiamante nella griglia

BLACS send/recv

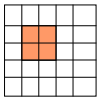
- yxxSD2D(ICONTXT, M, N, A, LDA, RDEST, CDEST)
- yxxRV2D(ICONTXT, M, N, A, LDA, RSRC, CSRC)

- Spedisce o riceve un blocco di matrice (rettangolare o trapezoidale)

Y (tipo di dati)	XX (tipo di matrice)
I: Intero S: Reale D: Doppia Prec. C: Complesso Z: Doppia prec. complessa	GE: matrice generale rettangolare TR: matrice trapezoidale

BLACS send/recv

- Esempio:
  - Matrice di ordine 5
  - Proc (0,0) spedisce a (1,1) il blocco A(2:3, 2:3) di ordine 2 di una matrice A di ordine 5



Matrice locale in possesso di (0,0)

```
IF( MYROW.EQ.0 .AND. MYCOL.EQ.0 ) THEN
    CALL DGESD2D( ICONTXT, 2, 2, A(2,2), 5, 1, 0 )
ELSE IF( MYROW.EQ.1 .AND. MYCOL.EQ.0 ) THEN
    CALL DGERV2D( ICONTXT, 2, 2, A(2,2), 5, 0, 0 )
END IF
```

BLACS: op. broadcast

- yxxBS2D(ICONTXT, SCOPE, TOP, M, N, A, LDA)
- yxxBR2D(ICONTXT, SCOPE, TOP, M, N, A, LDA, RSRC, CSRC)
- Broadcast di un blocco in un ambito (scope)
- SCOPE= 'Row', ' Column', 'All'
- Y e XX come send/recv

Esempio: broadcast da (0,0) sulla prima riga (riga 0) di processi

```
IF( MYROW.EQ.0 ) THEN
  IF( MYCOL.EQ.0 ) THEN
    CALL DGEBS2D( ICONTXT, 'Row', ' ', 2, 2, A, 3 )
  ELSE
    CALL DGEBR2D( ICONTXT, 'Row', ' ', 2, 2, A, 3, 0, 0 )
  END IF
END IF
```

BLACS: op. di riduzione

yGSUM2D(ICONTXT, SCOPE, TOP, M, N, A, LDA, RDEST, CDEST)

yGAMX2D(ICONTXT, SCOPE, TOP, M, N, A, LDA, RA, CA, RCFLAG, RDEST, CDEST)

yGAMN2D(ICONTXT, SCOPE, TOP, M, N, A, LDA, RA, CA, RCFLAG, RDEST, CDEST)

- Eseguono una operazione collettiva di riduzione (sum, max, min) in un ambito (scope).
- y come send/recv

PBLAS

- Parallel BLAS
- Libreria parallela a memoria distribuita per le operazioni di base dell'algebra lineare
- Sviluppata utilizzando
  - BLAS (per il calcolo)
  - BLACS (per la comunicazione)
- Obiettivi:
  - Leggibilita' (il codice e' piu' corto)
  - Efficienza (sfrutta l'efficienza di BLAS)
  - Modularita' (fornisce grandi building blocks per lo sviluppo di applicazioni)
  - Portabilita' (la dipendenza dalla macchina e' confinata in PBLAS)
- http://www.netlib.org/scalapack

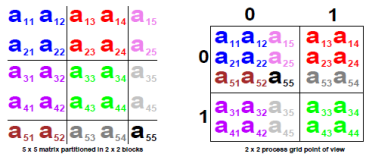
Memorizzazione delle matrici

- Una matrice di ordine MxN e' partizionata in blocchi di ordine MBxNB secondo la distribuzione ciclica a blocchi
- La matrice e' memorizzata per colonne (come BLAS)
- La matrice viene descritta da un array descrittore

DESCA(1): (DTYPE) 1
DESCA(2): (CTXT) contesto BLACS
DESCA(3): (M) numero di righe nella matrice globale
DESCA(4): (N) numero di colonne nella matrice globale
DESCA(5): (MB) numero di righe in un blocco
DESCA(6): (NB) numero di colonne in un blocco
DESCA(7): (RSRC) indice di riga del proprietario di A(1, 1)
DESCA(8): (CSRC) indice di colonna del proprietario di A(1, 1)
DESCA(9): (LLD) Leading dimension della matrice locale

Memorizzazione delle matrici

- Esempio:



DESCA(3): (M) = 5 numero di righe nella matrice globale
DESCA(4): (N) = 5numero di colonne nella matrice globale
DESCA(5): (MB) = 2 numero di righe in un blocco
DESCA(6): (NB) = 2 numero di colonne in un blocco
DESCA(7): (RSRC) =0 indice di riga del proprietario di A(1, 1)
DESCA(8): (CSRC) = 0 indice di colonna del proprietario di A(1, 1)
DESCA(9): (LLD) = { **3 sui proc della riga 0**; **2 sui proc della riga 1** } Leading dimension della matrice locale

Uso di PBLAS

Interfaccia molto simile a quella di BLAS

BLAS3

```
      CALL DGEMM( 'No Transpose', 'No Transpose',
     $    M-J-JB+1, N-J-JB+1, JB, -ONE, A(J+JB,J),
     $    LDA, A(J,J+JB), LDA, ONE, A(J+JB,J+JB), LDA )
```

PBLAS

```
      CALL PDGEMM( 'No Transpose', 'No Transpose',
     $    M-J-JB+1, N-J-JB+1, JB, -ONE, A, J+JB,
     $    J, DESCA, A, J, J+JB, DESCA, ONE, A,
     $    J+JB, J+JB, DESCA )
```

Da LAPACK a ScaLAPACK

```
      DO 20 J = 1, MIN( M, N ), NB
         JB = MIN( MIN( M, N )-J+1, NB )
*
*        Factor diagonal and subdiagonal blocks and test for exact  singularity.
*
         CALL DGETF2( M-J+1, JB, A( J, J ), LDA, IPIV( J ), IINFO )
*
*        Adjust INFO and the pivot indices.
*
         IF( INFO.EQ.0 .AND. IINFO.GT.0 )  INFO = IINFO + J - 1
         DO 10 I = J, MIN( M, J+JB-1 )
            IPIV( I ) = J - 1 + IPIV( I )
   10    CONTINUE
*
*        Apply interchanges to columns 1:J-1.
*
         CALL DLASWP( J-1, A, LDA, J, J+JB-1, IPIV, 1 )
*
         IF( J+JB.LE.N ) THEN
*
*           Apply interchanges to columns J+JB:N.
*
            CALL DLASWP( N-J-JB+1, A( 1, J+JB ), LDA, J, J+JB-1, IPIV, 1 )
*
*           Compute block row of U.
*
            CALL DTRSM( 'Left', 'Lower', 'No transpose', 'Unit', JB, N-J-JB+1, ONE, A( J, J ), LDA, A( J, J+JB ),  LDA )
            IF( J+JB.LE.M ) THEN
*
*              Update trailing submatrix.
*
               CALL DGEMM( 'No transpose', 'No transpose', M-J-JB+1, N-J-JB+1, JB, -ONE, A( J+JB, J ),
     $                     LDA, A( J, J+JB ), LDA, ONE, A( J+JB, J+JB ), LDA )
            END IF
         END IF
   20    CONTINUE
```

Routine DGETRF di **LAPACK** per la fatt. LU

Da LAPACK a ScaLAPACK

```
      DO 10 J = JN+1, JA+MN-1, DESCA( NB_ )
         JB = MIN( MN-J+JA, DESCA( NB_ ) )
         I = IA + J - JA
*
*        Factor diagonal and subdiagonal blocks and test for exact  singularity.
*
         CALL PDGETF2( M-J+JA, JB, A, I, J, DESCA, IPIV, IINFO )
*
         IF( INFO.EQ.0 .AND. IINFO.GT.0 )  INFO = IINFO + J - JA
*
*        Apply interchanges to columns JA:J-JA.
*
         CALL PDLASWP( 'Forward', 'Rowwise', J-JA, A, IA, JA, DESCA, I, I+JB-1, IPIV )
*
         IF( J-JA+JB+1.LE.N ) THEN
*
*           Apply interchanges to columns J+JB:JA+N-1.
*
            CALL PDLASWP( 'Forward', 'Rowwise', N-J-JB+JA, A, IA, J+JB,  DESCA, I, I+JB-1, IPIV )
*
*           Compute block row of U.
*
            CALL PDTRSM( 'Left', 'Lower', 'No transpose', 'Unit', JB, N-J-JB+JA, ONE, A, I, J, DESCA, A, I, J+JB, DESCA )
*
            IF( J-JA+JB+1.LE.M ) THEN
*
*              Update trailing submatrix.
*
               CALL PDGEMM( 'No transpose', 'No transpose', M-J-JB+JA, N-J-JB+JA, JB, -ONE, A, I+JB, J, DESCA, A,
     $                      I, J+JB, DESCA, ONE, A, I+JB, J+JB, DESCA )
*
            END IF
         END IF
*
   10 CONTINUE
```

Routine PDGETRF di **ScaLAPACK** per la fatt. LU

Esempio : risoluzione AX=B (A,B,X matrici)

```
.....
  CALL BLACS_GET( -1, 0, ICTXT )                          inizializza BLACS
  CALL BLACS_GRIDINIT( ICTXT, 'Row-major', NPROW, NPCOL ) inizializza griglia
  CALL BLACS_GRIDINFO( ICTXT, NPROW, NPCOL,               definisci MYROW , MYCOL
&                      MYROW, MYCOL )
.....
  CALL DESCINIT( DESCA, M, N, MB, NB, RSRC, CSRC, ICTXT,  definisci DESCA, DESCB
&                MXLLDA,INFO )
  CALL DESCINIT( DESCB, N, NRHS, NB, NBRHS, RSRC, CSRC,
&                ICTXT,MXLLDB, INFO )
.....
  CALL MATINIT( A, DESCA, B, DESCB )                      definisci matrici A, B
.....
  CALL PDGESV( N, NRHS, A, IA, JA, DESCA, IPIV, B, IB, JB, risolvi sistema con
&              DESCB, INFO )                              ScaLAPACK
.....
  CALL BLACS_GRIDEXIT( ICTXT )                            rimuovi griglia
  CALL BLACS_EXIT( 0 )                                    disattiva BLACS
```

hardware di riferimento

- ScaLAPACK e' progettato per sistemi MIMD a memoria distribuita
  - Sistemi MPP: CRAY T3, IBM BG, SGI Origin, Blade server
  - Cluster di workstation (omogenei)
  - Qualunque sistema dove e' presente MPI
- Modello di programmazione SPMD

Problemi:

1. Quanto grande puo' essere il sistema?
2. ScaLAPACK e' scalabile?

Scalabilita'

- Si e' detto che se
  1. La dimensione del problema rimane costante in ogni processore
  2. Efficienza rimane costante al cresere di P

L'algoritmo e' considerato scalabile

- Cioe' se la dimensione del problema cresce linearmente con il numero di processori, allora e' possibile mantenere elevati livelli di efficienza
- In pratica e' normale una leggera degradazione

Scalabilita'

- Un problema di algebra lineare che coinvolge matrici di ordine N la dimensione del problema cresce come N^2
- Al crescere del numero di processi P deve rimanere costante il rapporto

N^2/P

E' possibile dimostrare che le routine di ScaLAPACK sono scalabili in questo senso

Come ottenere alte prestazioni

- Alcuni suggerimenti dovuti all'esperienza:
  - Per una matrice di ordine MxN, utilizzare un numero di processi circa
    $P = MxN/10^6$
  - Questo evita di risolvere problemi troppo piccoli su molti processori
  - Ed evita di risolvere problemi troppo grandi riducendo il grado di parallelismo

Alte prestazioni su MPP

- Utilizzo di una efficiente distribuzione di dati
  - Blocchi quadrati : N/Nb ~ 64
  - Griglia quadrata : P = Q
- Utilizzo di BLAS ottimizzato
  - Intel mkl, AMD acl
  - ATLAS
  - Non la compilazione F77 del codice fortran di riferimento