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

/*
*  Purpose
*  =======
*
*  PDGEMM  performs one of the matrix-matrix operations
*
*     sub( C ) := alpha*op( sub( A ) )*op( sub( B ) ) + beta*sub( C ),
*
*  where
*
*     sub( C ) denotes C(IC:IC+M-1,JC:JC+N-1),  and, op( X )  is one  of
*     op( X ) = X   or   op( X ) = X'.
*
*  Thus, op( sub( A ) ) denotes A(IA:IA+M-1,JA:JA+K-1)  if TRANSA = 'N',
*                               A(IA:IA+K-1,JA:JA+M-1)' if TRANSA = 'T',
*                               A(IA:IA+K-1,JA:JA+M-1)' if TRANSA = 'C',
*
*  and,  op( sub( B ) ) denotes B(IB:IB+K-1,JB:JB+N-1)  if TRANSB = 'N',
*                               B(IB:IB+N-1,JB:JB+K-1)' if TRANSB = 'T',
*                               B(IB:IB+N-1,JB:JB+K-1)' if TRANSB = 'C',
*
*  Alpha and beta are scalars.  A, B and C are matrices;  op( sub( A ) )
*  is an  m by k submatrix,  op( sub( B ) )  is an  k by n submatrix and
*  sub( C ) is an m by n submatrix.
*
*  Notes
*  =====
*
*  A description  vector  is associated with each 2D block-cyclicly dis-
*  tributed matrix.  This  vector  stores  the  information  required to
*  establish the  mapping  between a  matrix entry and its corresponding
*  process and memory location.
*
*  In  the  following  comments,   the character _  should  be  read  as
*  "of  the  distributed  matrix".  Let  A  be a generic term for any 2D
*  block cyclicly distributed matrix.  Its description vector is DESC_A:
*
*  NOTATION         STORED IN       EXPLANATION
*  ---------------- --------------- ------------------------------------
*  DTYPE_A (global) DESCA[ DTYPE_ ] The descriptor type.
*  CTXT_A  (global) DESCA[ CTXT_  ] The BLACS context handle, indicating
*                                   the NPROW x NPCOL BLACS process grid
*                                   A  is  distributed over. The context
*                                   itself  is  global,  but  the handle
*                                   (the integer value) may vary.
*  M_A     (global) DESCA[ M_     ] The  number of rows in the distribu-
*                                   ted matrix A, M_A >= 0.
*  N_A     (global) DESCA[ N_     ] The number of columns in the distri-
*                                   buted matrix A, N_A >= 0.
*  IMB_A   (global) DESCA[ IMB_   ] The number of rows of the upper left
*                                   block of the matrix A, IMB_A > 0.
*  INB_A   (global) DESCA[ INB_   ] The  number  of columns of the upper
*                                   left   block   of   the   matrix   A,
*                                   INB_A > 0.
*  MB_A    (global) DESCA[ MB_    ] The blocking factor used to  distri-
*                                   bute the last  M_A-IMB_A  rows of A,
*                                   MB_A > 0.
*  NB_A    (global) DESCA[ NB_    ] The blocking factor used to  distri-
*                                   bute the last  N_A-INB_A  columns of
*                                   A, NB_A > 0.
*  RSRC_A  (global) DESCA[ RSRC_  ] The process row over which the first
*                                   row of the matrix  A is distributed,
*                                   NPROW > RSRC_A >= 0.
*  CSRC_A  (global) DESCA[ CSRC_  ] The  process column  over  which the
*                                   first column of  A  is  distributed.
*                                   NPCOL > CSRC_A >= 0.
*  LLD_A   (local)  DESCA[ LLD_   ] The  leading dimension  of the local
```

```
*                                  array  storing  the  local blocks of
*                                  the distributed matrix A,
*                                  IF( Lc( 1, N_A ) > 0 )
*                                     LLD_A >= MAX( 1, Lr( 1, M_A ) )
*                                  ELSE
*                                     LLD_A >= 1.
*
*  Let K be the number of  rows of a matrix A starting at the global in-
*  dex IA,i.e, A( IA:IA+K-1, : ). Lr( IA, K ) denotes the number of rows
*  that the process of row coordinate MYROW ( 0 <= MYROW < NPROW ) would
*  receive if these K rows were distributed over NPROW processes.  If  K
*  is the number of columns of a matrix  A  starting at the global index
*  JA, i.e, A( :, JA:JA+K-1, : ), Lc( JA, K ) denotes the number  of co-
*  lumns that the process MYCOL ( 0 <= MYCOL < NPCOL ) would  receive if
*  these K columns were distributed over NPCOL processes.
*
*  The values of Lr() and Lc() may be determined via a call to the func-
*  tion PB_Cnumroc:
*  Lr( IA, K ) = PB_Cnumroc( K, IA, IMB_A, MB_A, MYROW, RSRC_A, NPROW )
*  Lc( JA, K ) = PB_Cnumroc( K, JA, INB_A, NB_A, MYCOL, CSRC_A, NPCOL )
*
*  Arguments
*  =========
*
*  TRANSA  (global input) CHARACTER*1
*          On entry,  TRANSA  specifies the form of op( sub( A ) ) to be
*          used in the matrix multiplication as follows:
*
*             TRANSA = 'N' or 'n'   op( sub( A ) ) = sub( A ),
*
*             TRANSA = 'T' or 't'   op( sub( A ) ) = sub( A )',
*
*             TRANSA = 'C' or 'c'   op( sub( A ) ) = sub( A )'.
*
*  TRANSB  (global input) CHARACTER*1
*          On entry,  TRANSB  specifies the form of op( sub( B ) ) to be
*          used in the matrix multiplication as follows:
*
*             TRANSB = 'N' or 'n'   op( sub( B ) ) = sub( B ),
*
*             TRANSB = 'T' or 't'   op( sub( B ) ) = sub( B )',
*
*             TRANSB = 'C' or 'c'   op( sub( B ) ) = sub( B )'.
*
*  M       (global input) INTEGER
*          On entry,  M  specifies  the number of rows of the  submatrix
*          op( sub( A ) ) and of the submatrix sub( C ). M  must  be  at
*          least  zero.
*
*  N       (global input) INTEGER
*          On entry, N specifies the number of columns of the  submatrix
*          op( sub( B ) )  and  the  number of columns of the  submatrix
*          sub( C ). N must be at least zero.
*
*  K       (global input) INTEGER
*          On entry, K specifies the number of columns of the  submatrix
*          op( sub( A ) )  and  the  number of rows   of  the  submatrix
*          op( sub( B ) ). K must be at least  zero.
*
*  ALPHA   (global input) DOUBLE PRECISION
*          On entry, ALPHA specifies the scalar alpha.   When  ALPHA  is
*          supplied  as zero then the local entries of the arrays  A and
*          B corresponding to the entries of  the  submatrices  sub( A )
*          and sub( B ) respectively need not be set on input.
*
*  A       (local input) DOUBLE PRECISION array
*          On entry, A is an array of dimension (LLD_A, Ka), where Ka is
*          at least Lc( 1, JA+K-1 ) when  TRANSA = 'N' or 'n', and is at
*          least  Lc( 1, JA+M-1 )  otherwise.  Before  entry, this array
*          contains the local entries of the matrix A.
```

```
*
*  IA      (global input) INTEGER
*          On entry, IA  specifies A's global row index, which points to
*          the beginning of the submatrix sub( A ).
*
*  JA      (global input) INTEGER
*          On entry, JA  specifies A's global column index, which points
*          to the beginning of the submatrix sub( A ).
*
*  DESCA   (global and local input) INTEGER array
*          On entry, DESCA  is an integer array of dimension DLEN_. This
*          is the array descriptor for the matrix A.
*
*  B       (local input) DOUBLE PRECISION array
*          On entry, B is an array of dimension (LLD_B, Kb), where Kb is
*          at least Lc( 1, JB+N-1 ) when  TRANSB = 'N' or 'n', and is at
*          least Lc( 1, JB+K-1 )  otherwise.  Before  entry,  this array
*          contains the local entries of the matrix B.
*
*  IB      (global input) INTEGER
*          On entry, IB  specifies B's global row index, which points to
*          the beginning of the submatrix sub( B ).
*
*  JB      (global input) INTEGER
*          On entry, JB  specifies B's global column index, which points
*          to the beginning of the submatrix sub( B ).
*
*  DESCB   (global and local input) INTEGER array
*          On entry, DESCB  is an integer array of dimension DLEN_. This
*          is the array descriptor for the matrix B.
*
*  BETA    (global input) DOUBLE PRECISION
*          On entry,  BETA  specifies the scalar  beta.   When  BETA  is
*          supplied  as  zero  then  the  local entries of  the array  C
*          corresponding to  the  entries of the submatrix sub( C ) need
*          not be set on input.
*
*  C       (local input/local output) DOUBLE PRECISION array
*          On entry, C is an array of dimension (LLD_C, Kc), where Kc is
*          at least Lc( 1, JC+N-1 ).  Before  entry, this array contains
*          the local entries of the matrix  C.
*          On exit, the entries of this array corresponding to the local
*          entries of the  submatrix  sub( C )  are  overwritten  by the
*          local entries of the m by n updated submatrix.
*
*  IC      (global input) INTEGER
*          On entry, IC  specifies C's global row index, which points to
*          the beginning of the submatrix sub( C ).
*
*  JC      (global input) INTEGER
*          On entry, JC  specifies C's global column index, which points
*          to the beginning of the submatrix sub( C ).
*
*  DESCC   (global and local input) INTEGER array
*          On entry, DESCC  is an integer array of dimension DLEN_. This
*          is the array descriptor for the matrix C.
*
*  -- Written on April 1, 1998 by
*     Antoine Petitet, University of Tennessee, Knoxville 37996, USA.
*
*  ---------------------------------------------------------------------
*/
```