root

Go Up

Name	PBblas
Version	3.0.2
Description	Parallel Block Basic Linear Algebra Subsystem
License	http://www.apache.org/licenses/LICENSE-2.0
Copyright	Copyright (C) 2016, 2017 HPCC Systems
Authors	HPCCSystems
DependsOn	ML_Core
Platform	6.2.0

OVERVIEW

PBblas

Parallel BLAS support for ECL Machine Learning library This is a high-performance, scalable version of the Basic Linear Algebra Subsystem. This is used by many of the Machine-Learning bundles

Table of Contents

Module to convert between ML_Core/Types Field layouts (i.e
ExtractTri.ecl Extract the upper or lower triangle from the composite output from getrf (LU Factorization)
gemm.ecl Extended Parallel Block Matrix Multiplication Module
getrf.ecl Perform LU Factoriztion of a Matrix
HadamardProduct.ecl Element-wise multiplication of X * Y
IElementFunc.ecl Function prototype for a function to apply to each element of the distributed matrix using Apply2Elements
MatUtils.ecl Provides various Utility attributes for manipulating cell-based matrixes
potrf.ecl Produce a Cholesky factorization of a matrix
scal.ecl Scale a matrix by a constant
tran.ecl Transpose a matrix and (optionally) add a second matrix
trsm.ecl Partitioned block parallel triangular matrix solver
Types.ecl Types for the Parallel Block Basic Linear Algebra Sub-programs support
Vector2Diag.ecl Convert a vector into a diagonal matrix

Apply2Elements

Go Up

IMPORTS

_versions.PBblas.V3_0_2.PBblas.Types | std.blas |

DESCRIPTIONS

APPLY2ELEMENTS Apply2Elements

/ EXPORT DATASET(Layout_Cell)	**Apply2Elements**
(DATASET(Layout_Cell) X, IElementFunc f)	

Apply a user-defined function to each element of the matrix.

Use PBblas.IElementFunc as the prototype function. Input and output may be a single matrix, or myriad matrixes with different work item ids.

PARAMETER **X** ||| TABLE (Layout_Cell) — A matrix (or multiple matrices) in Layout_Cell form.

PARAMETER **f** ||| FUNCTION [REAL8 , UNSIGNED4 , UNSIGNED4] (REAL8) — A function based on the IElementFunc prototype.

RETURN **TABLE ({ UNSIGNED2 wi_id , UNSIGNED4 x , UNSIGNED4 y , REAL8 v })** — A matrix (or multiple matrices) in Layout_Cell form.

SEE PBblas/IElementFunc

SEE PBblas/Types.Layout_Cell

---

asum

Go Up

IMPORTS

_versions.PBblas.V3_0_2.PBblas.Types | _versions.PBblas.V3_0_2.PBblas.internal | _versions.PBblas.V3_0_2.PBblas.internal.Types | _versions.PBblas.V3_0_2.PBblas.internal.MatDims | _versions.PBblas.V3_0_2.PBblas.internal.Converted | std.blas |

DESCRIPTIONS

ASUM asum

`/ EXPORT DATASET(Layout_Norm)`	**asum**
`(DATASET(Layout_Cell) X)`	

Calculate the absolute sum – the ”Entrywise” 1-norm of a matrix.

Compute SUM(ABS(X)).

PARAMETER **X** ||| TABLE (Layout_Cell) — Matrix or set of matrices in Layout_Cell format.

RETURN **TABLE ({ UNSIGNED2 wi_id , REAL8 v })** — DATASET(Layout_Norm) with one record per work item.

SEE PBblas/Types.Layout_Cell

axpy

Go Up

IMPORTS

_versions.PBblas.V3_0_2.PBblas.Types |

DESCRIPTIONS

AXPY axpy

/ EXPORT DATASET(Layout_Cell)	axpy
(value_t alpha, DATASET(Layout_Cell) X, DATASET(Layout_Cell) Y)	

Scale a matrix and add a second matrix.

Implements alpha*X + Y.

X and Y must have same shape.

PARAMETER **alpha** ||| REAL8 — Scalar multiplier for the X matrix.

PARAMETER **X** ||| TABLE (Layout_Cell) — X matrix in DATASET(Layout_Cell) form.

PARAMETER **Y** ||| TABLE (Layout_Cell) — Y matrix in DATASET(Layout_Call) form.

RETURN **TABLE ({ UNSIGNED2 wi_id , UNSIGNED4 x , UNSIGNED4 y , REAL8 v })** — Matrix in DATASET(Layout_Cell) form.

SEE PBblas/Types.Layout_Cell

Converted

Go Up

IMPORTS

_versions.PBblas.V3_0_2.PBblas.Types |

_versions.ML_Core.V3_2_2.ML_Core.Types |

DESCRIPTIONS

CONVERTED Converted

	Converted

Module to convert between ML_Core/Types Field layouts (i.e. NumericField and DiscreteField) and PBblas matrix layout (i.e. Layout_Cell).

ML_Core and PBblas use different forms to represent numeric matrices.

ML_Core utilizes two forms:

- NumericField – Real-valued matrix.
- DiscreteField – Discrete-valued (Integer) matrix.

PBblas uses the Layout_Cell format.

While both the ML_Core form and the PBblas form represent matrices, there are different semantics implied. The ML_Core matrices are used to represent a series of observations (rows), each with multiple features (columns). The PBblas Layout_Cell represents a matrix of rows and columns with no further semantic meaning implied.

Children

1. NFToMatrix : Convert NumericField dataset to PBblas Layout_Cell dataset
2. DFToMatrix : Convert DiscreteField dataset to PBblas Matrix
3. MatrixToNF : Convert PBblas Matrix to NumericField dataset
4. MatrixToDF : Convert PBblas Matrix to DiscreteField dataset

---

NFTOMATRIX NFToMatrix

Converted \

`DATASET(Layout_Cell)`	**NFToMatrix**
`(DATASET(NumericField) recs)`	

Convert NumericField dataset to PBblas Layout_Cell dataset.

PARAMETER **recs** ||| TABLE (NumericField) — Record Dataset in DATASET(NumericField) format.

RETURN **TABLE ({ UNSIGNED2 wi_id , UNSIGNED4 x , UNSIGNED4 y , REAL8 v })** — Matrix in DATASET(Layout_Cell) format.

SEE PBblas/Types.Layout_Cell

SEE ML_Core/Types.NumericField

DFTOMATRIX DFToMatrix

Converted \

DATASET(Layout_Cell)	**DFToMatrix**
(DATASET(DiscreteField) recs)	

Convert DiscreteField dataset to PBblas Matrix.

PARAMETER **recs** ||| TABLE (DiscreteField) — Record Dataset in DATASET(DiscreteField) format.

RETURN **TABLE ({ UNSIGNED2 wi_id , UNSIGNED4 x , UNSIGNED4 y , REAL8 v })** — Matrix in DATASET(Layout_Cell) format.

SEE PBblas/Types.Layout_Cell

SEE ML_Core/Types.DiscreteField

MATRIXTONF MatrixToNF

Converted \

DATASET(NumericField)	**MatrixToNF**
(DATASET(Layout_Cell) mat)	

Convert PBblas Matrix to NumericField dataset.

PARAMETER **mat** ||| TABLE (Layout_Cell) — Matrix in DATASET(Layout_Cell) format.

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED8 id , UNSIGNED4 number , REAL8 value })** — NumericField Dataset.

SEE PBblas/Types.Layout_Cell

SEE ML_Core/Types.NumericField

MATRIXTODF MatrixToDF

Converted \

DATASET(DiscreteField)	**MatrixToDF**
(DATASET(Layout_Cell) mat)	

Convert PBblas Matrix to DiscreteField dataset.

PARAMETER <u>**mat**</u> ||| TABLE (Layout_Cell) — Matrix in DATASET(Layout_Cell) format.

RETURN **TABLE ({ UNSIGNED2 wi , UNSIGNED8 id , UNSIGNED4 number , INTEGER4 value })** — DiscreteField Dataset.

SEE PBblas/Types.Layout_Cell

SEE ML_Core/Types.DiscreteField

ExtractTri

Go Up

IMPORTS

std.blas | _versions.PBblas.V3_0_2.PBblas.Types |

_versions.PBblas.V3_0_2.PBblas.internal |

_versions.PBblas.V3_0_2.PBblas.internal.Types |

_versions.PBblas.V3_0_2.PBblas.internal.MatDims |

_versions.PBblas.V3_0_2.PBblas.internal.Converted |

DESCRIPTIONS

EXTRACTTRI ExtractTri

`/ EXPORT DATASET(Layout_Cell)`	**ExtractTri**
`(Triangle tri, Diagonal dt, DATASET(Layout_Cell) A)`	

Extract the upper or lower triangle from the composite output from getrf (LU Factorization).

PARAMETER **tri** ||| UNSIGNED1 — Triangle type: Upper or Lower (see Types.Triangle).

PARAMETER **<u>dt</u>** ||| UNSIGNED1 — Diagonal type: Unit or non unit (see Types.Diagonal).

PARAMETER **<u>A</u>** ||| TABLE (Layout_Cell) — Matrix of cells. See Types.Layout_Cell.

RETURN **TABLE ({ UNSIGNED2 wi_id , UNSIGNED4 x , UNSIGNED4 y , REAL8 v })** — Matrix of cells in Layout_Cell format representing a triangular matrix (upper or lower).

SEE PBblas.Types

---

gemm

Go Up

IMPORTS

_versions.PBblas.V3_0_2.PBblas.Types | _versions.PBblas.V3_0_2.PBblas.internal | _versions.PBblas.V3_0_2.PBblas.internal.Types | std.blas | _versions.PBblas.V3_0_2.PBblas.internal.MatDims | std.system.Thorlib |

DESCRIPTIONS

GEMM gemm

/ EXPORT DATASET(Layout_Cell)	**gemm**
(BOOLEAN transposeA, BOOLEAN transposeB, value_t alpha, DATASET(Layout_Cell) A_in, DATASET(Layout_Cell) B_in, DATASET(Layout_Cell) C_in=emptyC, value_t beta=0.0)	

Extended Parallel Block Matrix Multiplication Module.

Implements: alpha * op(A) * op(B) + beta * C. op is No Transpose or Transpose.

Multiplies two matrixes A and B, with an optional pre-multiply transpose for each.

Optionally scales the product by the scalar ”alpha”.

Then adds an optional C matrix to the product after scaling C by the scalar "beta".

A, B, and C are specified as DATASET(Layout_Cell), as is the Resulting matrix. Layout_Cell describes a sparse matrix stored as a list of x, y, and value.

This interface also provides a "Myriad" capability allowing multiple similar operations to be performed on independent sets of matrixes in parallel. This is done by use of the work-item id (wi_id) in each cell of the matrixes.

Cells with the same wi_id are considered part of the same matrix.

In the myriad form, each input matrix A, B, and (optionally) C can contain many independent matrixes. The wi_ids are matched up such that each operation involves the A, B, and C with the same wi_id. A and B must therefore contain the same set of wi_ids, while C is optional for any wi_id. The same parameters: alpha, beta, transposeA, and transposeB are used for all work-items.

The result will contain cells from all provided work-items.

Result has same shape as C if provided. Note that matrixes are not explicitly dimensioned. The shape is determined by the highest value of x and y for each work-item.

PARAMETER **transposeA** ||| BOOLEAN — Boolean indicating whether matrix A should be transposed before multiplying.

PARAMETER **transposeB** ||| BOOLEAN — Same as above but for matrix B.

PARAMETER **alpha** ||| REAL8 — Scaling factor for the A matrix.

PARAMETER **A_in** ||| TABLE (Layout_Cell) — 'A' matrix (multiplier) in Layout_Cell format.

PARAMETER **B_in** ||| TABLE (Layout_Cell) — Same as above for the 'B' matrix (multiplicand).

PARAMETER **C_in** ||| TABLE (Layout_Cell) — Same as above for the 'C' matrix (addend). May be omitted.

PARAMETER **beta** ||| REAL8 — A scalar multiplier for beta * C, scales the C matrix before addition. May be omitted.

RETURN **TABLE ({ UNSIGNED2 wi_id , UNSIGNED4 x , UNSIGNED4 y , REAL8 v })** — Result matrix in Layout_Cell format.

SEE PBblas/Types.Layout_Cell

---

getrf

Go Up

IMPORTS

_versions.PBblas.V3_0_2.PBblas.Types | _versions.PBblas.V3_0_2.PBblas.internal |

_versions.PBblas.V3_0_2.PBblas.internal.Types | std.blas |

_versions.PBblas.V3_0_2.PBblas.internal.MatDims | std.system.Thorlib |

DESCRIPTIONS

GETRF getrf

`/ EXPORT DATASET(Layout_Cell)`	**getrf**
`(DATASET(Layout_Cell) A)`	

Perform LU Factoriztion of a Matrix.

Splits a matrix into Lower and Upper triangular factors

Produces composite LU matrix for the diagonal blocks.

Iterate through the matrix a row of blocks and column of blocks at a time. Partition A into M block rows and N block columns. The A11 cell is a single block. A12 is a single row of blocks with N-1 columns. A21 is a single column of blocks with M-1 rows. A22 is a sub-matrix of M-1 x N-1 blocks.

```
| A11   A12 |      |  L11   0   |    | U11        U12    |
| A21   A22 |  ==  |  L21   L22 | *  |  0         U22    |

                   | L11*U11            L11*U12          |
              ==   | L21*U11        L21*U12 + L22*U22    |
```

Based upon PB-BLAS: A set of parallel block basic linear algebra subprograms by Choi and Dongarra

This module supports the ”Myriad” style interface, allowing many independent problems to be worked on at once. The A matrix can contain multiple matrixes to be factored, indicated by different values for work-item id (wi_id).

Note: The returned matrix includes both the upper and lower factors. This matrix can be used directly by trsm which will only use the part indicated by trsm’s ’triangle’ parameter (i.e. upper or lower). To extract the upper or lower triangle explicitly for other purposes, use the ExtractTri function. When passing the Lower matrix to the triangle solver (trsm), set the ”Diagonal” parameter to ”UnitTri”. This is necessary because both triangular matrixes returned from this function are packed into a square matrix with only one diagonal. By convention, The Lower triangle is assumed to be a Unit Triangle (diagonal all ones), so the diagonal contained in the returned matrix is for the Upper factor and must be ignored (i.e. assumed to be all ones) when referencing the Lower triangle.

PARAMETER **A** ||| TABLE (Layout_Cell) — The input matrix in Types.Layout_Cell format.

RETURN **TABLE ({ UNSIGNED2 wi_id , UNSIGNED4 x , UNSIGNED4 y , REAL8 v })** — Resulting factored matrix in Layout_Cell format.

SEE Types.Layout_Cell

SEE ExtractTri

HadamardProduct

Go Up

IMPORTS

_versions.PBblas.V3_0_2.PBblas.internal |

_versions.PBblas.V3_0_2.PBblas.internal.MatDims |

_versions.PBblas.V3_0_2.PBblas.Types |

_versions.PBblas.V3_0_2.PBblas.internal.Types |

_versions.PBblas.V3_0_2.PBblas.internal.Converted | std.blas | std.system.Thorlib |

DESCRIPTIONS

HADAMARDPRODUCT HadamardProduct

/ EXPORT DATASET(Layout_Cell)	**HadamardProduct**
(DATASET(Layout_Cell) X, DATASET(Layout_Cell) Y)	

Element-wise multiplication of X * Y.

Result[x,y] := X[x,y] * Y[x,y].

Supports the "myriad" style interface – X and Y may contain multiple separate matrixes. Each X will be multiplied by the Y with the same work-item id.

Note: This performs element-wise multiplication. For dot-product matrix multiplication, use PBblas.gemm.

PARAMETER **X** ||| TABLE (Layout_Cell) — A matrix (or multiple matrices) in Layout_Cell form.

PARAMETER **Y** ||| TABLE (Layout_Cell) — A matrix (or multiple matrices) in Layout_Cell form.

RETURN **TABLE ({ UNSIGNED2 wi_id , UNSIGNED4 x , UNSIGNED4 y , REAL8 v })** — A matrix (or multiple matrices) in Layout_Cell form.

SEE PBblas/Types.Layout_Cell

IElementFunc

Go Up

IMPORTS

DESCRIPTIONS

IELEMENTFUNC IElementFunc

/ EXPORT value_t	**IElementFunc**
(value_t v, dimension_t r, dimension_t c)	

Function prototype for a function to apply to each element of the distributed matrix using Apply2Elements.

Base your cell-wise function on this prototype.

PARAMETER **v** ||| REAL8 — Input value.

PARAMETER **r** ||| UNSIGNED4 — Row number (1 based).

PARAMETER **c** ||| UNSIGNED4 — Column number (1 based).

RETURN **REAL8** — Output value.

SEE PBblas/Apply2Elements

MatUtils

Go Up

IMPORTS

_versions.PBblas.V3_0_2.PBblas.Types | _versions.PBblas.V3_0_2.PBblas.internal | _versions.PBblas.V3_0_2.PBblas.internal.Types | _versions.PBblas.V3_0_2.PBblas.internal.MatDims |

DESCRIPTIONS

MATUTILS MatUtils

	MatUtils

Provides various Utility attributes for manipulating cell-based matrixes.

SEE Std/PBblas/Types.Layout_Cell

Children

1. GetWorkItems : Get a list of work-item ids from a matrix containing one or more work items
2. InsertCols : Insert one or more columns of a fixed value into a matrix
3. Transpose : Transpose a matrix

GETWORKITEMS GetWorkItems

MatUtils \

`DATASET(Layout_WI_ID)`	**GetWorkItems**
`(DATASET(Layout_Cell) cells)`	

Get a list of work-item ids from a matrix containing one or more work items.

PARAMETER **cells** ||| TABLE (Layout_Cell) — A matrix in Layout_Cell format.

RETURN **TABLE ({ UNSIGNED2 wi_id })** — DATASET(Layout_WI_ID), one record per work-item.

SEE PBblas/Types.Layout_Cell

SEE PBblas/Types.Layout_WI_ID

INSERTCOLS InsertCols

MatUtils \

`DATASET(Layout_Cell)`	**InsertCols**
`(DATASET(Layout_Cell) M, UNSIGNED cols_to_insert=1, value_t insert_val=1)`	

Insert one or more columns of a fixed value into a matrix.

Columns are inserted before the first original column.

This attribute supports the myriad interface. Multiple independent matrixes can be represented by M.

PARAMETER M ||| TABLE (Layout_Cell) — the input matrix in Layout_Cell format.

PARAMETER cols_to_insert ||| UNSIGNED8 — the number of columns to insert, default 1.

PARAMETER insert_val ||| REAL8 — the value for each cell of the new column(s), default 0.

RETURN **TABLE ({ UNSIGNED2 wi_id , UNSIGNED4 x , UNSIGNED4 y , REAL8 v })** — matrix in Layout_Cell format with additional column(s)

TRANSPOSE Transpose

MatUtils \

`DATASET(Layout_Cell)`	**Transpose**
`(DATASET(Layout_Cell) M)`	

Transpose a matrix.

This attribute supports the myriad interface. Multiple independent matrixes can be represented by M.

PARAMETER M ||| TABLE (Layout_Cell) — A matrix in DATASET(Layout_Cell) format.

RETURN **TABLE ({ UNSIGNED2 wi_id , UNSIGNED4 x , UNSIGNED4 y , REAL8 v })** — Transposed matrix in Layout_Cell format.

SEE PBblas/Types.Layout_Cell

---

potrf

Go Up

IMPORTS

_versions.PBblas.V3_0_2.PBblas.Types | std.blas |

_versions.PBblas.V3_0_2.PBblas.internal |

_versions.PBblas.V3_0_2.PBblas.internal.Types |

_versions.PBblas.V3_0_2.PBblas.internal.MatDims |

_versions.PBblas.V3_0_2.PBblas.internal.Converted | std.system.Thorlib |

DESCRIPTIONS

POTRF potrf

`/ EXPORT DATASET(Layout_Cell)`	**potrf**
`(Triangle tri, DATASET(Layout_Cell) A_in)`	

Produce a Cholesky factorization of a matrix.

Cholesky factorization of A such that A = U**T * U if Triangular.Upper requested or A = L * L**T if Triangualr.Lower is requested.

Note that the Cholesky factorization in Linear Algebra is analogous to a square-root in scalar algebra.

The matrix A must be symmetric positive definite.

```
 | A11   A12 |       |  L11   0    |     | L11**T     L21**T |
 | A21   A22 |  ==   |  L21   L22 | *   |  0           L22   |

                     | L11*L11**T          L11*L21**T        |
                ==   | L21*L11**T  L21*L21**T + L22*L22**T |

So, use Cholesky on the first block to get L11.
    L21 = A21*L11**T**-1   which can be found by dtrsm on each column block
    A22' is A22 - L21*L21**T
```

Based upon PB-BLAS: A set of parallel block basic linear algebra subprograms by Choi and Dongarra

This module supports the "Myriad" style interface, allowing many independent problems to be worked on at once. The A matrix can contain multiple matrixes to be factored, indicated by different values for work-item id (wi_id).

PARAMETER **tri** ||| UNSIGNED1 — Types.Triangle enumeration indicating whether we are looking for the Upper or the Lower factor.

PARAMETER **A_in** ||| TABLE (Layout_Cell) — The matrix or matrixes to be factored in Types.Layout_Cell format.

RETURN **TABLE ({ UNSIGNED2 wi_id , UNSIGNED4 x , UNSIGNED4 y , REAL8 v })** — Triangular matrix in Layout_Cell format.

SEE Types.Layout_Cell

SEE Types.Triangle

---

scal

Go Up

IMPORTS

_versions.PBblas.V3_0_2.PBblas.Types |

DESCRIPTIONS

SCAL scal

/ EXPORT DATASET(Layout_Cell)	scal
(value_t alpha, DATASET(Layout_Cell) X)	

Scale a matrix by a constant.

Result is alpha * X

This supports a "myriad" style interface in that X may be a set of independent matrices separated by different work-item ids.

PARAMETER **alpha** ||| REAL8 — A scalar multiplier.

PARAMETER **X** ||| TABLE (Layout_Cell) — The matrix(es) to be scaled in Layout_Cell format.

RETURN **TABLE ({ UNSIGNED2 wi_id , UNSIGNED4 x , UNSIGNED4 y , REAL8 v })** — Matrix in Layout_Cell form, of the same shape as X.

SEE PBblas/Types.Layout_Cell

---

tran

Go Up

IMPORTS

_versions.PBblas.V3_0_2.PBblas.Types | _versions.PBblas.V3_0_2.PBblas.internal | _versions.PBblas.V3_0_2.PBblas.internal.Types | _versions.PBblas.V3_0_2.PBblas.internal.MatDims | _versions.PBblas.V3_0_2.PBblas.internal.Converted | std.blas | std.system.Thorlib |

DESCRIPTIONS

TRAN **tran**

`DATASET(Layout_Cell)`	**tran**
`(value_t alpha, DATASET(Layout_Cell) A, value_t beta=0, DATASET(Layout_Cell) C=empty_c)`	

Transpose a matrix and (optionally) add a second matrix.

Implements: result <== alpha * A**t + beta * C, A is n by m, C is m by n

A**T (A Transpose) and C must have same shape.

PARAMETER **alpha** ||| REAL8 — Scalar multiplier for the A**T matrix.

PARAMETER **A** ||| TABLE (Layout_Cell) — A matrix in DATASET(Layout_Cell) form.

PARAMETER **beta** ||| REAL8 — (Optional) Scalar multiplier for the C matrix.

PARAMETER **C** ||| TABLE (Layout_Cell) — (Optional) C matrix in DATASET(Layout_Call) form.

RETURN **TABLE ({ UNSIGNED2 wi_id , UNSIGNED4 x , UNSIGNED4 y , REAL8 v })** — Resulting matrix in DATASET(Layout_Cell) form.

SEE Types.layout_cell

---

trsm

Go Up

IMPORTS

_versions.PBblas.V3_0_2.PBblas.Types | std.blas |

_versions.PBblas.V3_0_2.PBblas.internal |

_versions.PBblas.V3_0_2.PBblas.internal.Types |

_versions.PBblas.V3_0_2.PBblas.internal.MatDims |

_versions.PBblas.V3_0_2.PBblas.internal.Converted | std.system.Thorlib |

DESCRIPTIONS

TRSM trsm

`/ EXPORT DATASET(Layout_Cell)`	**trsm**
`(Side s, Triangle tri, BOOLEAN transposeA, Diagonal diag, value_t alpha, DATASET(Layout_Cell) A_in, DATASET(Layout_Cell) B_in)`	

Partitioned block parallel triangular matrix solver.

Solves for X using: AX = B or XA = B.

A is is a triangular matrix, X and B have the same dimensions.

A may be an upper triangular matrix (UX = B or XU = B), or a lower triangular matrix (LX = B or XL = B).

Allows optional transposing and scaling of A.

Partially based upon an approach discussed by MJ DAYDE, IS DUFF, AP CERFACS. A Parallel Block implementation of Level-3 BLAS for MIMD Vector Processors ACM Tran. Mathematical Software, Vol 20, No 2, June 1994 pp 178-193 and other papers about PB-BLAS by Choi and Dongarra.

This module supports the ”Myriad” style interface, allowing many independent problems to be worked on at once. Corresponding A and B matrixes are related by a common work-item identifier (wi_id) within each cell of the matrix. The returned X matrix will contain cells for the same set of work-items as specified for the A and B matrices.

PARAMETER **s** ||| UNSIGNED1 — Types.Side enumeration indicating whether we are solving AX = B or XA = B

PARAMETER **tri** ||| UNSIGNED1 — Types.Triangle enumeration indicating whether we are solving an Upper or Lower triangle.

PARAMETER **transposeA** ||| BOOLEAN — Boolean indicating whether or not to transpose the A matrix before solving.

PARAMETER **diag** ||| UNSIGNED1 — Types.Diagonal enumeration indicating whether A is a unit matrix or not. This is primarily used after factoring matrixes using getrf (LU factorization). That module produces a factored matrix stored within the same space as the original matrix. Since the diagonal is used by both factors, by convention, the Lower triangle has a unit matrix (diagonal all 1's) while the Upper triangle uses the diagonal cells. Setting this to UnitTri, causes the contents of the diagonal to be ignored, and assumed to be 1. NotUnitTri should be used for most other cases.

PARAMETER **alpha** ||| REAL8 — Multiplier to scale A.

PARAMETER **A_in** ||| TABLE (Layout_Cell) — The A matrix in Layout_Cell format.

PARAMETER **B_in** ||| TABLE (Layout_Cell) — The B matrix in Layout_Cell format.

RETURN **TABLE ({ UNSIGNED2 wi_id , UNSIGNED4 x , UNSIGNED4 y , REAL8 v })** — X solution matrix in Layout_Cell format.

SEE Types.Layout_Cell

SEE Types.Triangle

SEE Types.Side

Types

Go Up

IMPORTS

_versions.ML_Core.V3_2_2.ML_Core |

_versions.ML_Core.V3_2_2.ML_Core.Types |

DESCRIPTIONS

TYPES Types

	Types

Types for the Parallel Block Basic Linear Algebra Sub-programs support.

WARNING: attributes marked with WARNING can not be changed without making corresponding changes to the C++ attributes.

Children

1. dimension_t : Type for matrix dimensions
2. partition_t : Type for partition id – only supports up to 64K partitions
3. work_item_t : Type for work-item id – only supports up to 64K work items
4. value_t : Type for matrix cell values
5. m_label_t : Type for matrix label

6. Triangle : Enumeration for Triangle type WARNING: type used in C++ attribute
7. Diagonal : Enumeration for Diagonal type WARNING: type used in C++ attribute
8. Side : Enumeration for Side type in trsm
9. Layout_Cell : Layout for a Matrix Cell
10. Layout_Norm : Layout for Norm results

DIMENSION_T dimension_t

Types \

	dimension_t

Type for matrix dimensions. Uses UNSIGNED4 as matrixes are not designed to support more than 4 B rows or columns.

RETURN **UNSIGNED4** —

PARTITION_T partition_t

Types \

	partition_t

Type for partition id – only supports up to 64K partitions.

RETURN **UNSIGNED2** —

WORK_ITEM_T work_item_t

Types \

	work_item_t

Type for work-item id – only supports up to 64K work items.

RETURN **UNSIGNED2** —

VALUE_T value_t

Types \

	value_t

Type for matrix cell values WARNING: type used in C++ attribute

RETURN **REAL8** —

M_LABEL_T m_label_t

Types \

	m_label_t

Type for matrix label. Used for Matrix dimensions (see Layout_Dims) and for partitions (see Layout_Part).

RETURN **STRING3** —

TRIANGLE Triangle

Types \

	Triangle

Enumeration for Triangle type WARNING: type used in C++ attribute.

RETURN **UNSIGNED1** —

VALUE Upper = 1

VALUE Lower = 2

DIAGONAL Diagonal

Types \

	Diagonal

Enumeration for Diagonal type WARNING: type used in C++ attribute.

RETURN **UNSIGNED1** —

VALUE UnitTri = 1. Ignore the values of the diagonal and use all ones instead.

VALUE NotUnitTri = 2. Use the diagonal values.

SIDE Side

Types \

	Side

Enumeration for Side type in trsm. WARNING: type used in C++ attribute

RETURN **UNSIGNED1** —

VALUE Ax = 1. Solve x for Ax = B.

VALUE xA = 2. Solve x for xA = B.

SEE trsm

LAYOUT_CELL Layout_Cell

Types \

	Layout_Cell

Layout for a Matrix Cell.

Main representation of Matrix cell at interface to all PBBlas functions.

Matrixes are represented as DATASET(Layout_Cell), where each cell describes the row and column position of the cell as well as its value. Only the non-zero cells need to be contained in the dataset in order to describe the matrix since all unspecified cells are considered to have a value of zero. The cell also contains a work-item number that allows multiple separate matrixes to be carried in the same dataset. This supports the ”myriad” style interface that allows the same operations to be performed on many different sets of data at once.

Note that these matrixes do not have an explicit size. They are sized implicitly, based on the maximum row and column presented in the data.

A matrix can be converted to an explicit dense form (see matrix_t) by using the utility module MakeR8Set. That module should only be used for known small matrixes (< 1M cells) or for partitions of a larger matrix.

The 'internal/Converted' module provides utility functions to convert to and from a set of partitions used internally (See Layout_parts).

WARNING: Used as C++ attribute. Do not change without corresponding changes to MakeR8Set.

FIELD **wi_id** ||| UNSIGNED2 — Work Item Number – An identifier from 1 to 64K-1 that separates and identifies individual matrixes.

FIELD **x** ||| UNSIGNED4 — 1-based row position within the matrix.

FIELD **y** ||| UNSIGNED4 — 1-based column position within the matrix.

FIELD **v** ||| REAL8 — Real value for the cell.

SEE matrix_t

SEE MakeR8Set.ecl

SEE internal/Converted.ecl

LAYOUT_NORM Layout_Norm

Types \

	Layout_Norm

Layout for Norm results.

FIELD **wi_id** ||| UNSIGNED2 — Work Item Number – An identifier from 1 to 64K-1 that separates and identifies individual matrixes

FIELD **v** ||| REAL8 — Real value for the norm

Vector2Diag

Go Up

IMPORTS

_versions.PBblas.V3_0_2.PBblas.internal |

_versions.PBblas.V3_0_2.PBblas.internal.MatDims |

_versions.PBblas.V3_0_2.PBblas.Types |

_versions.PBblas.V3_0_2.PBblas.internal.Types |

_versions.PBblas.V3_0_2.PBblas.Constants |

DESCRIPTIONS

VECTOR2DIAG Vector2Diag

`/ EXPORT DATASET(Layout_Cell)`	**Vector2Diag**
`(DATASET(Layout_Cell) X)`	

Convert a vector into a diagonal matrix.

The typical notation is D = diag(V).

The input X must be a 1 x N column vector or an N x 1 row vector.

The resulting matrix, in either case will be N x N, with zero everywhere except the diagonal.

PARAMETER **X** ||| TABLE (Layout_Cell) — A row or column vector (i.e. N x 1 or 1 x N) in Layout_Cell format.

RETURN **TABLE ({ UNSIGNED2 wi_id , UNSIGNED4 x , UNSIGNED4 y , REAL8 v })** — An N x N matrix in Layout_Cell format.

SEE Types.Layout_cell
