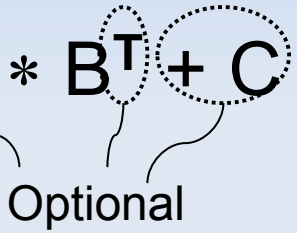
# GML – Global Matrix Library Overview

***by***

*Juemin Zhang*(*Jeff*) @ **IBM**

# Computational Function

- Cell-wise operations
  - Add, subtract, multiply, division
  - Sum, Eculeadian distance, etc
- Matrix multiply
  - $C = A^T * B^T + C$




```
C.mult(A, B, addOn);
C.transMult(A, B, addOn);
C.multTrans(A, B, addOn);
C.transMultTans(A, B, addOn);
```

- Result is stored in invoking object, which is also the return object
  - Chained operations

# Matrix Type

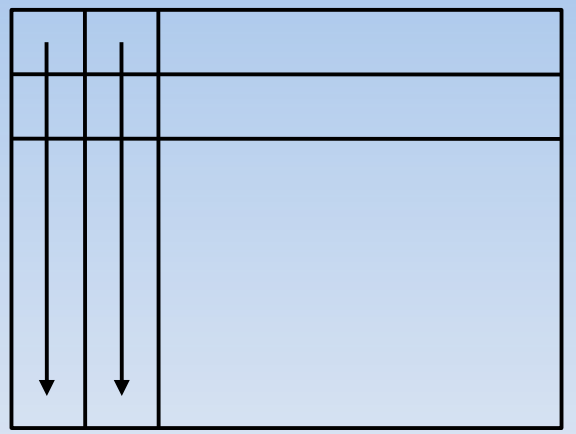
- Dense matrix
- Sparse matrix
- Block matrix
- Distributed dense matrix
- Distributed sparse matrix
- Duplicated dense matrix
- Duplicated sparse matrix

# Matrix methods

- Constructor
  - Matrix dimensions, storage, block partitioning, block distribution, sparsity
- Make
  - Memory allocation
- Clone, and CopyTo
- Initialization
- Matrix element access this(x,y) and set
- Matrix subset: column(s) and row(s)
- toString, print

# Dense Matrix

- Memory layout
  - Column-major memory storage
- BLAS driver (double precision)
  - Level 1, 2, and 3
  - Dense-dense operation
- X10 driver
  - Dense-dense
  - Dense-sparse
- Result is stored in dense



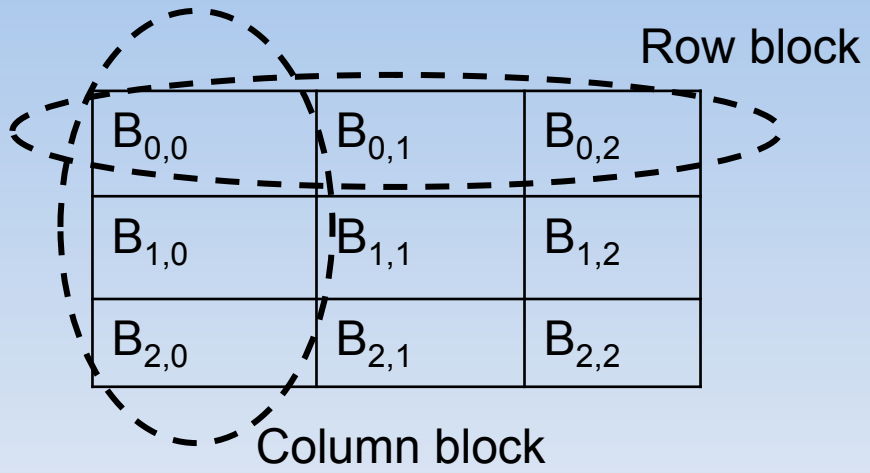
```
0  for (var c:Int=0; c<B.N; c++) {
1      for (var k:Int=0; k<A.N; k++) {
2          val v2    = B(k, c);
3          for (var r:Int=0; r<A.M; r++) {
4              val v1= A(r, k);
5              C.d(startcol+r) = v1 * v2;
6          }
7          startcol += C.M;
8      }
   }
```

# Sparse Matrix

- Compressed sparse column
  - *Matrix elements, each of which contains index and nonzero value pair, are is read first by column*
- 2D compressed array
  - 1D compressed array
- Compress array
  - Index array
  - Value array
- Not good for inserting or modification
  - *Sparse matrix is not used to store matrix computation result*

# Matrix Blocks and Block Matrix

- Matrix partitioning - Grid




- Matrix blocks
  - Row block ID and column block ID
  - Dense/Sparse matrix
- Block matrix
  - Number of row blocks, number of column blocks
  - List of rows for all row blocks, and list of columns for all column blocks
  - List of dense/sparse blocks

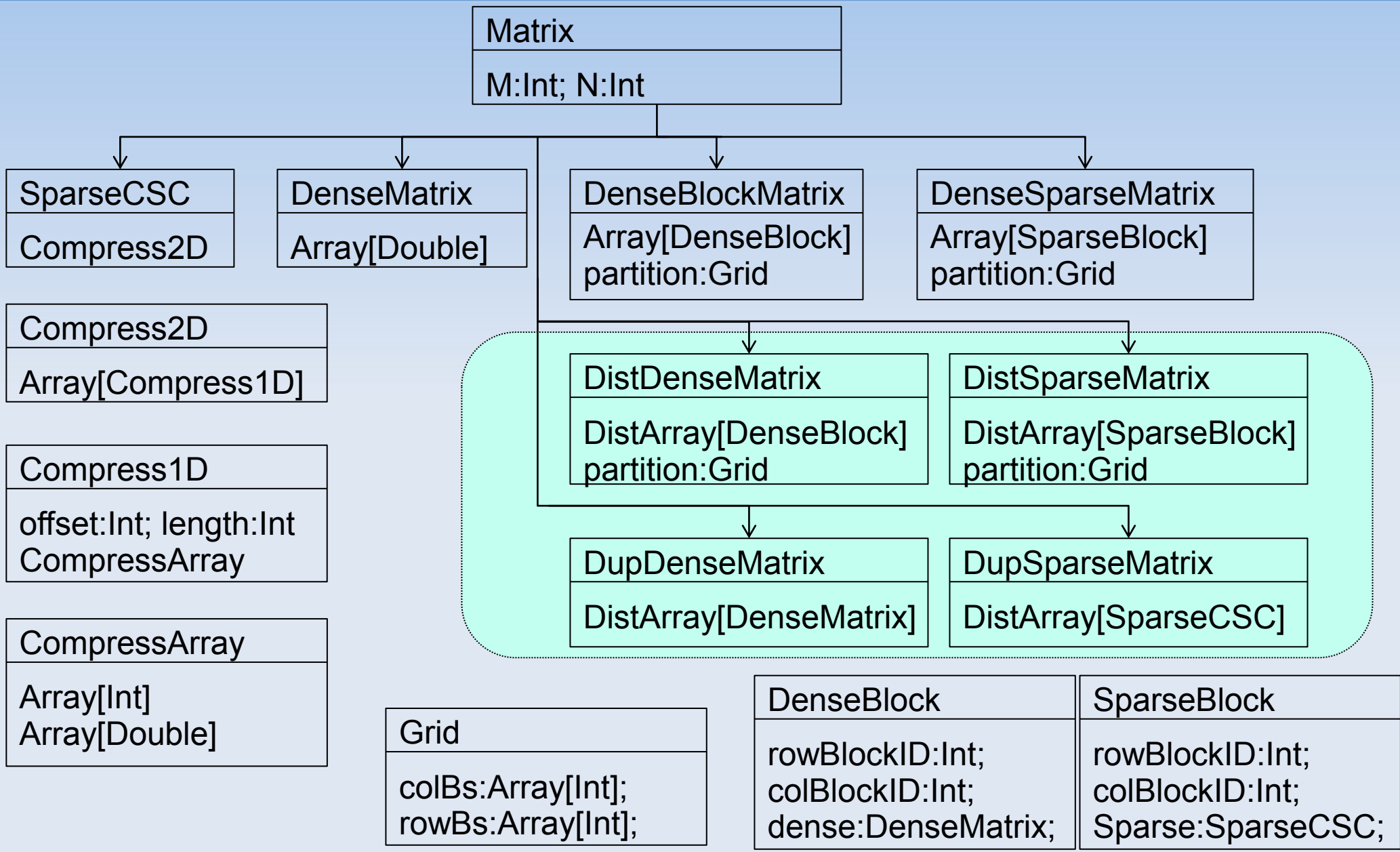
# Distributed and Duplicated Matrix

- Distributed dense/sparse matrix
  - Matrix blocks unique distributed on DistArray

| | | |
|---|---|---|
| $Place_0$ $B_{0,0}$ | $Place_3$ $B_{0,1}$ | $Place_6$ $B_{0,2}$ |
| $Place_1$ $B_{1,0}$ | $Place_4$ $B_{1,1}$ | $Place_7$ $B_{1,2}$ |
| $Place_2$ $B_{2,0}$ | $Place_5$ $B_{2,1}$ | $Place_8$ $B_{2,2}$ |

- Duplicated dense/sparse matrix
  - Replicated matrices are uniquely distributed at all places on DistArray

# Matrix Class Structure



# Access data from different place

- X10 implicit data capture
  - At (..) { }
  - Expensive, but easy to use

| | |
|---|---|
| Data declare | **val data:…..;** |
| | **…..** |
| Remote execution block | **at ( … ) {** |
| | **…;** |
| Remote data capture | **… data …;** |
| | **}** |

- Explicit data copy
  - Remote array copy
    - LocalPlaceHandle, array on DistArray and dense/sparse on DistArray
    - Portable to different transports: sockets, lapi and pami
  - MPI wrapped calls (only for native backend on MPI transport)
    - Array on DistArray and dense/sparse on DistArray
    - Optimized collective operations
  - Fast, but difficult in programming

# Inter-place data communication

- Place-to-place communication
- Collective communication
  - Bcast:
    - Synchronize copies of duplicated matrix
  - Scatter:
    - Copy data from block matrix to distributed one
    - Copy data from single-column/row matrix to distributed one
    - Copy data from single row block matrix to distributed one
  - Gather
    - Reverse of scatter
  - Reduce
    - Sum of all blocks

# Build GML

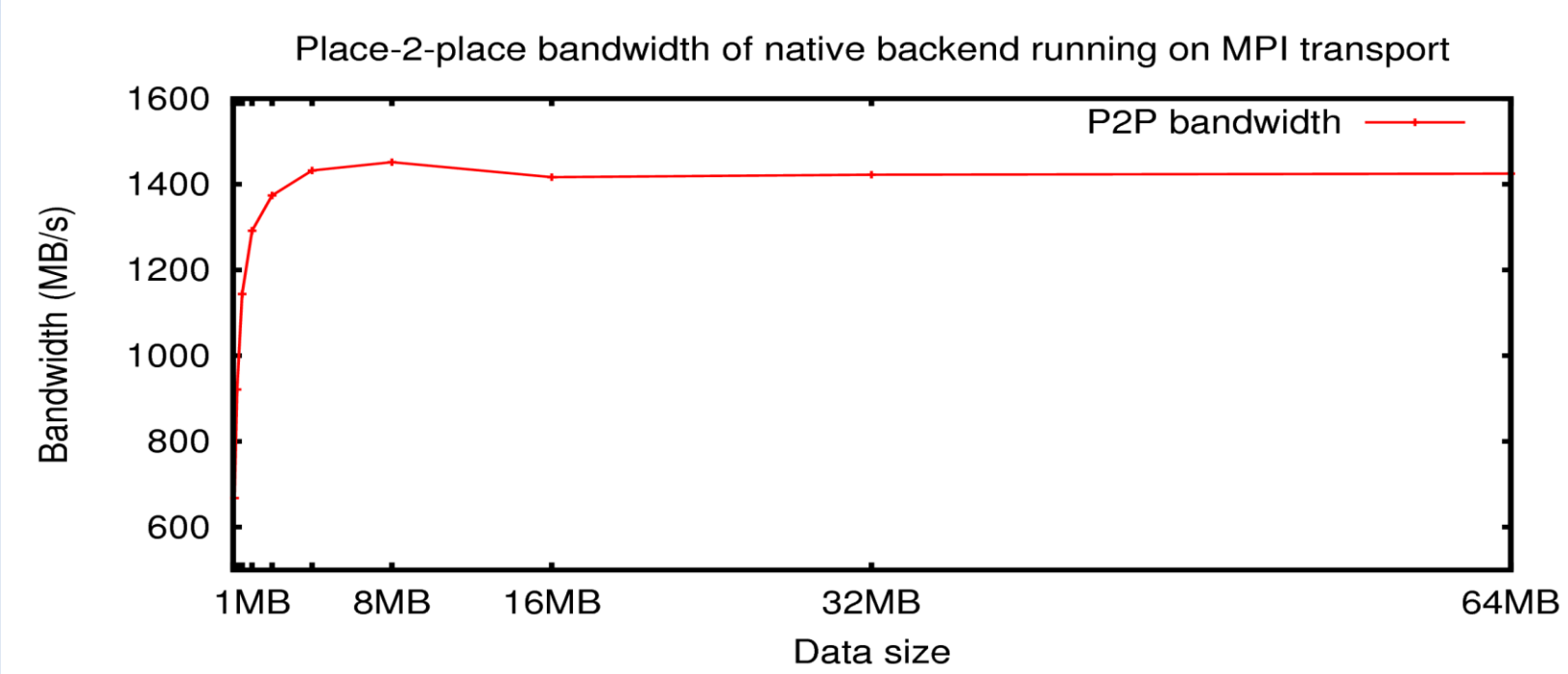
- Setup “system_setting.mk”
  - X10 compiler path
  - MPI/g++
  - BLAS path
- Build GML library for native/managed backend
  - Native backend: make native
  - Native backend + MPI transport: make native_mpi
  - Managed backend: make managed
- Build application
  - Add: -classpath [gml_path]/[**managed**|**native**|**native_mpi**]_gml.jar
  - Add: -x10lib [gml_path]/[**managed**|**native native_mpi**]_gml.properties

# Benchmark platform - Triloka

- 10-node cluster
  - Node: 2 Quad-Core AMD Opteron(tm) 2356
- Memory: 16-GB each node
- Network
  - Infinitband (MPI): 20 Gb/sec (4X DDR)
  - Ethernet (socket):1Gb/sec

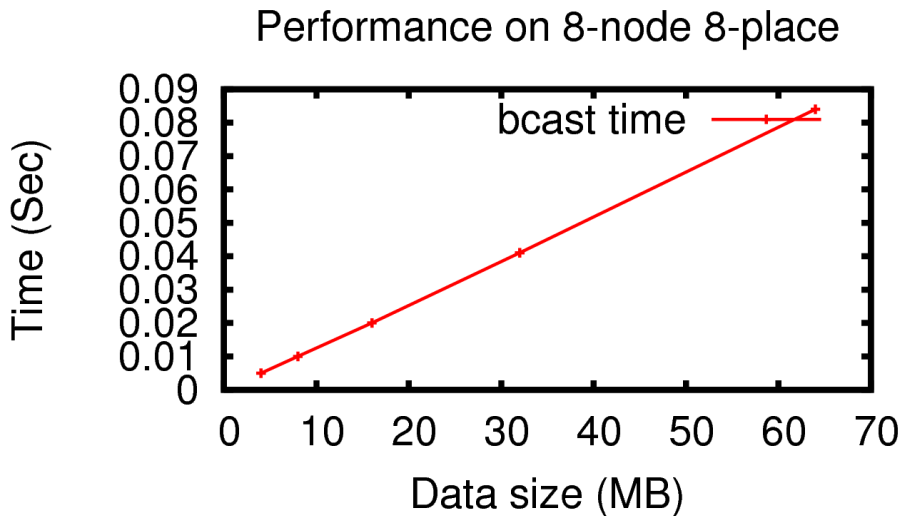
# Communication Benchmark Place-to-Place

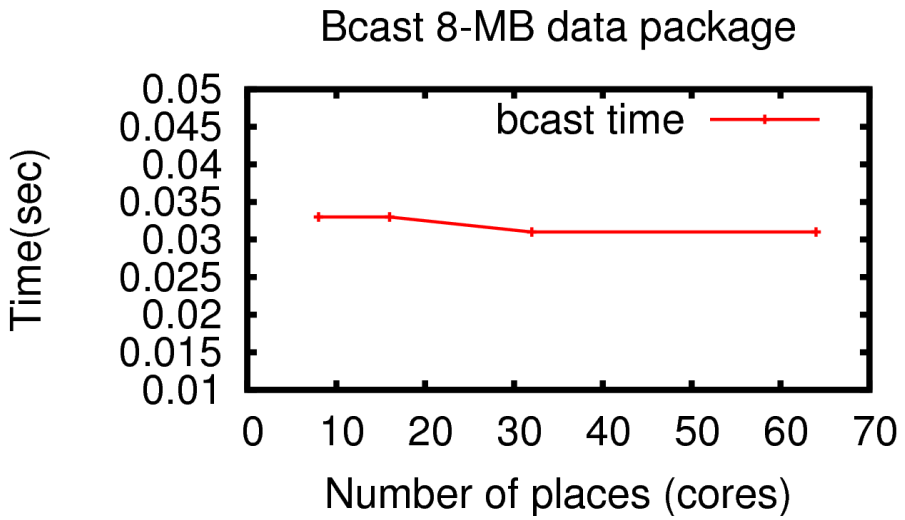
- Native backend running on MPI transport (InfiniBand 20Gb/s)
- X10 Peak performance: 1.45GB/s
  - Intel MPI benchmark peak performance 1.48GB/s



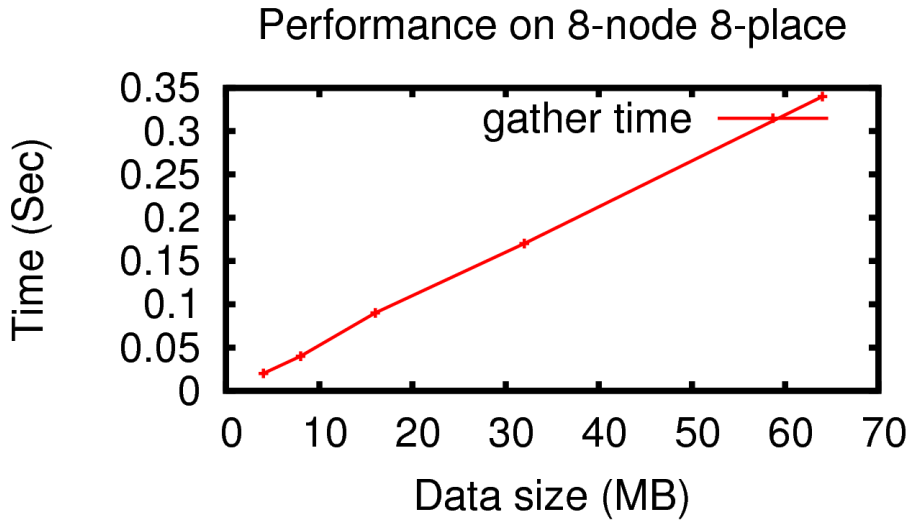
# Collective Communication Benchmarks: bcast, gather

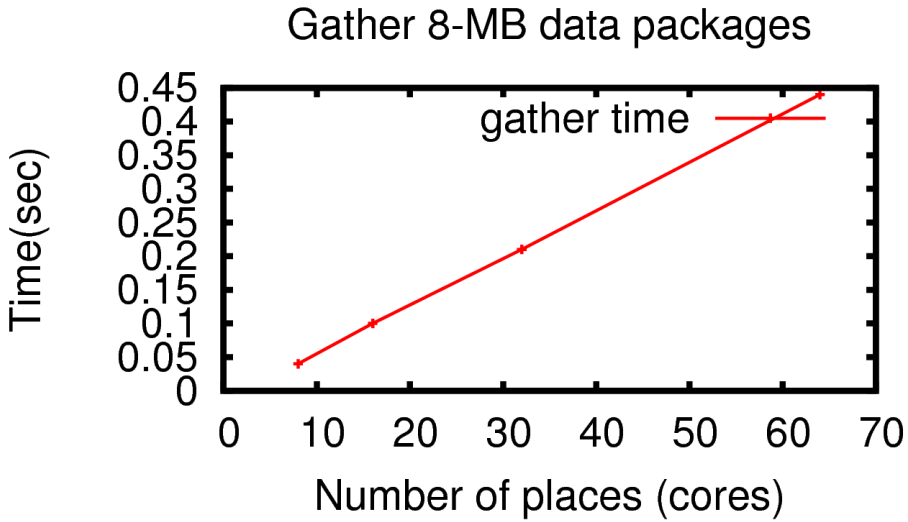
Bcast benchmark of native backend running on MPI-transport



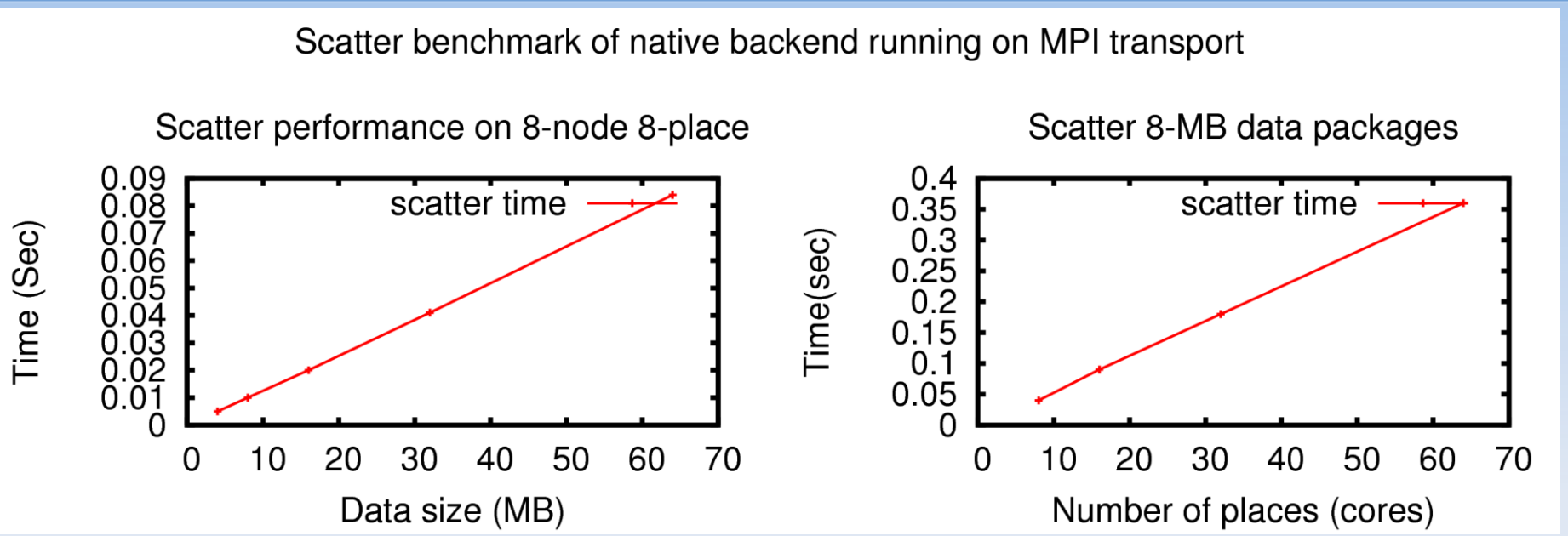



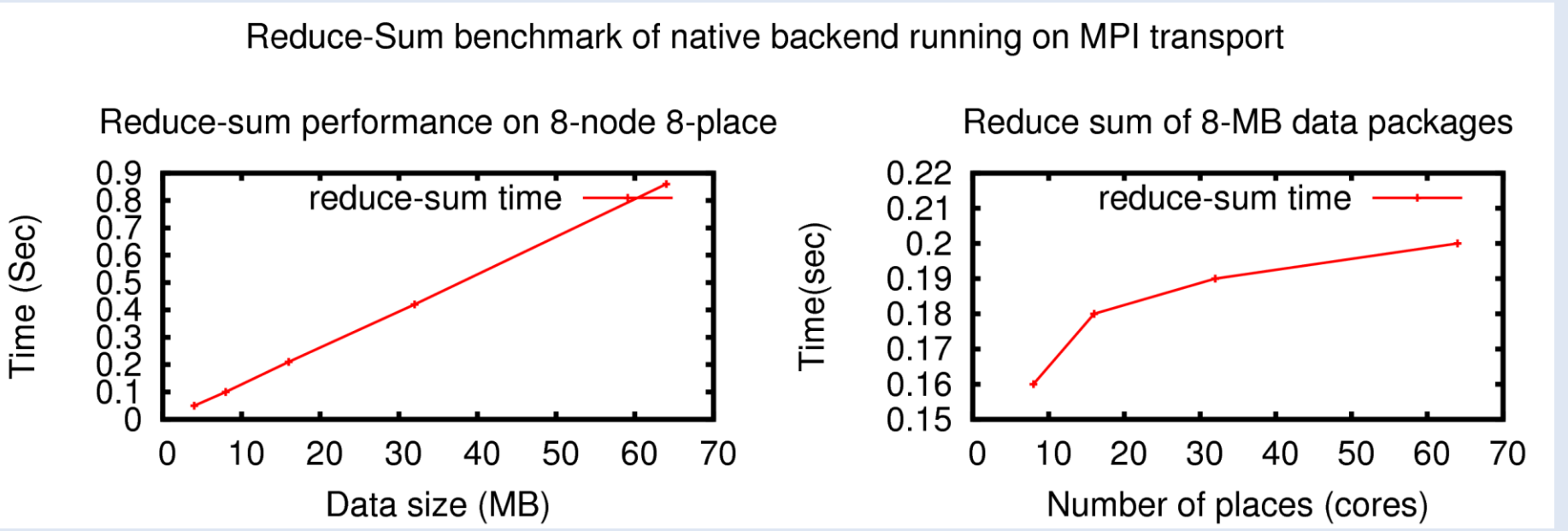

Gather benchmark of native backend running on MPI transport



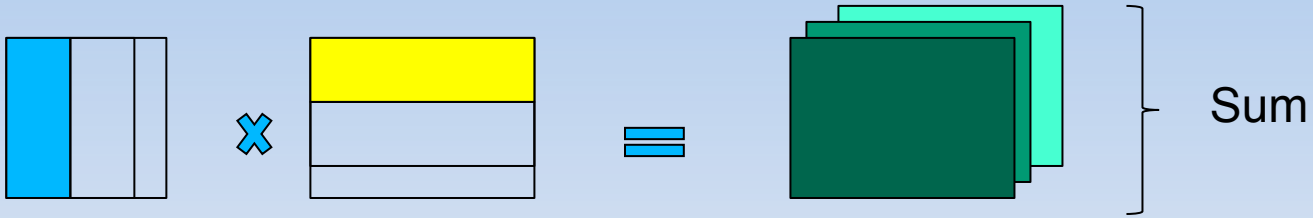
# Collective Communication Benchmarks: scatter, reduce



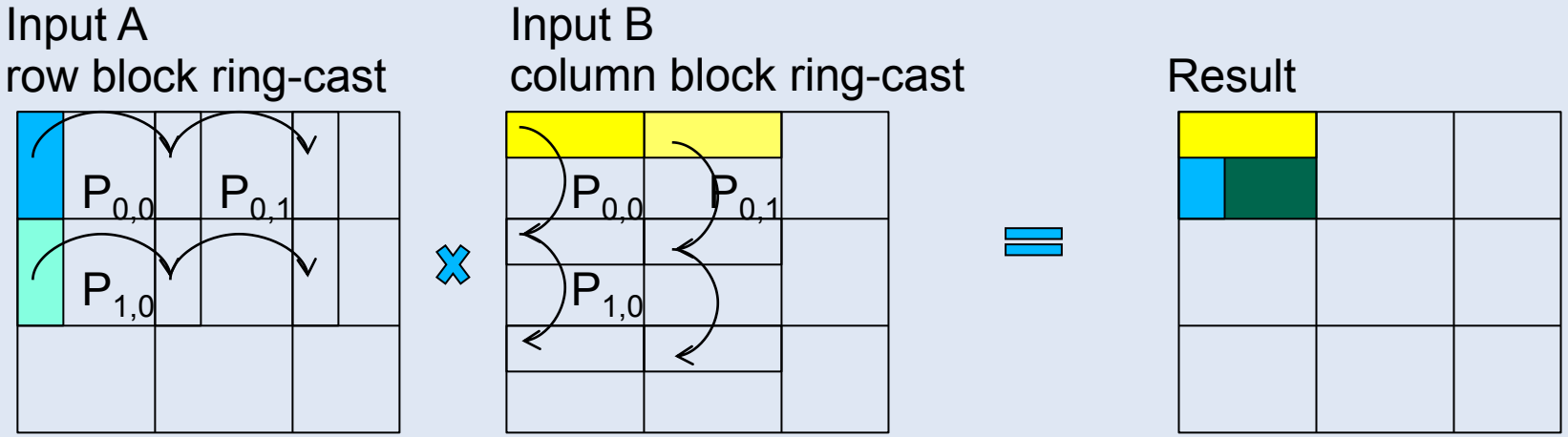



# SUMMA A x B

- Local (in place) matrix * matrix



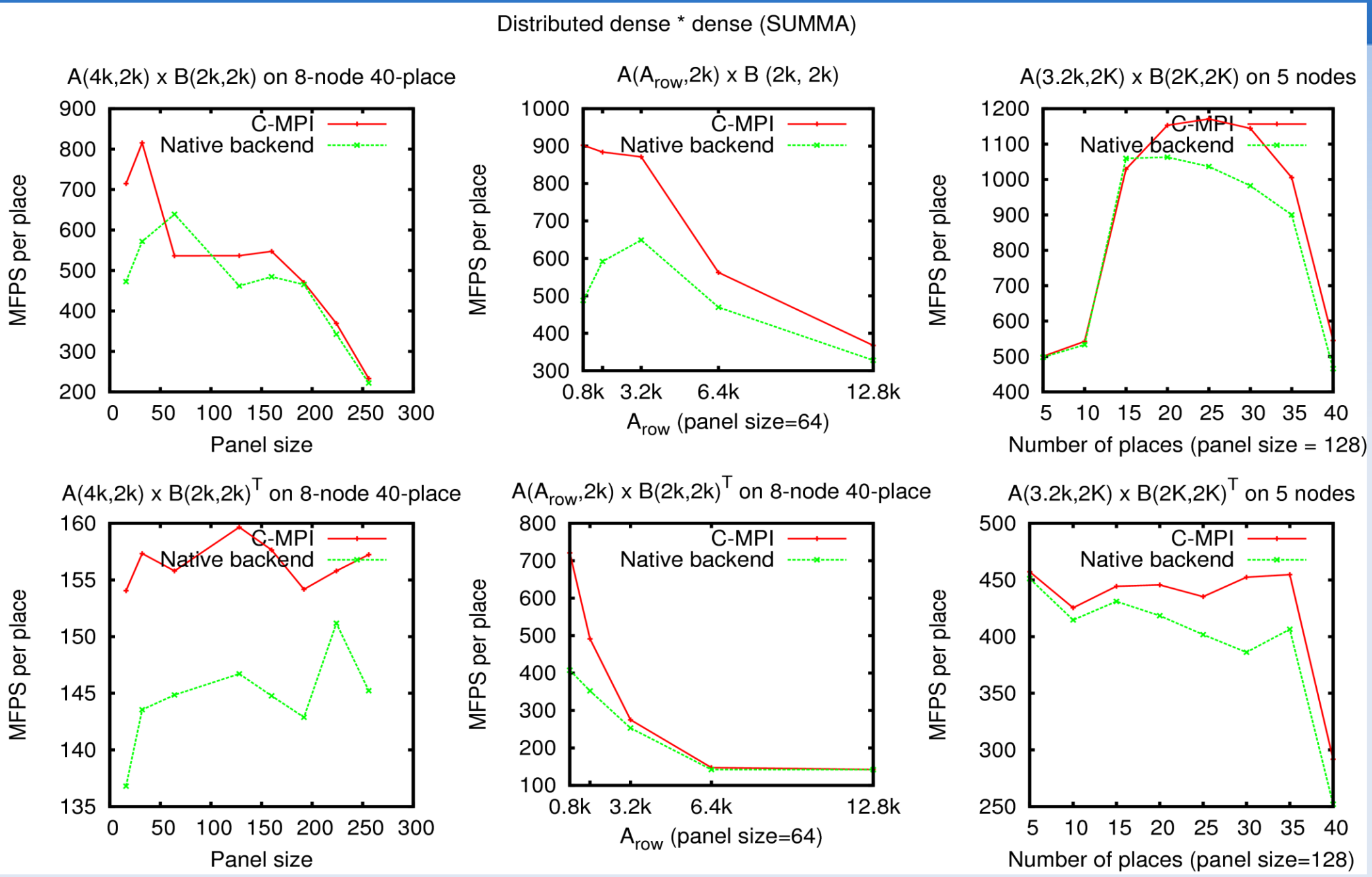

- Matrices are partitioned in the same way



# Test Settings

- DistDenseMatrix SUMMA
  - Computation: BLAS library
  - Comparing:
    - C-MPI SUMMA implementation
    - X10 SUMMA implementation native backend running on MPI transport
- DistSparseMatrix SUMMA
  - Computation: X10 sparse*sparse driver

# DistDenseMatrix SUMMA



# SUMMA (DistSparseMatrix)

Sparse * sparse (density 0.01) SUMMA benchmark of native backend on MPI transport