Parallel Matrix Distribution Library for Sparse Matrix Solvers

Akihiro Fujii (Kogakuin Univ.)
Reiji Suda (Tokyo Univ.)
Akira Nishida (Tokyo Univ.)
Target Problem

- Parallel matrix distribution library for sparse matrix solvers which can deal with
  - Load-balancing
  - Ordering of the distributed matrix
- Usability of the library
  - Easiness to use and to create various load-balancing methods
Outline

- Background and Objective
- Matrix Distribution Library
  - Library Interface
- Load-balancing Examples using the library
- Numerical Tests of the examples
- Evaluation and Conclusion
- Future Work
Background and Objective

- Few libraries enable users to distribute the sparse matrix
  - Because user’s data structure cannot be specified to one matrix format.
- Few libraries manage the ordering of the rows of the distributed matrix.

- Especially for sparse matrix solvers, the CRS format is often used

Objective: Consider and implement such library based on the distributed CRS format.
Features of the Matrix Distribution Library

- Any distribution and any order of the rows of the matrix can be specified for re-distribution.
- The library doesn’t use the particular matrix format other than distributed CRS format.
  - So users can use the distributed matrix easily.
  - If necessary, re-distribution can be done repeatedly.
- Memory is managed by users.
  - Users know how much memory is needed to reorder and redistribute the matrix before the matrix distribution.
Library interface which specifies both distribution and ordering

<table>
<thead>
<tr>
<th>PE0</th>
<th>PE1</th>
<th>PE2</th>
<th>PE3</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>6</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

order(·) shows new row number for each row after reordering

index_number(·) has the last row numbers on PEs after distribution

Allocated row numbers after distribution

order(·) shows new row number for each row after reordering

index_number(·) has the last row numbers on PEs after distribution
Implementation

- Copy-and-Delete strategy is taken.
- Execution steps as follows.
  - The user call the library subroutine to calculate the size of the new reordered matrix, and then allocate the memory.
  - The library redistribute the sparse matrix there.
Library Subroutines

- **set_order(some arguments..)**
  - stores the Ordering information in the arrays of the arguments.

- **size_new_mat(some arguments..)**
  - calculates the size of new distributed matrix size.

- **assign_new_mat(some arguments..)**
  - has the arguments of user allocated space, and assigns new distributed matrix there.

- **distribute_vector(some arguments..)**
  - distributes vector in the way specified in the arguments.
Various Matrix Distribution using this library as examples

The case: According to the weight on each PE, the sparse matrix is distributed

- Distribution with lowest communication cost
  - lowest comm
- Distribution based on ParMETIS
  -metis dist
- Distribution with no reordering
  - constant order
Distribution method 1: lowest comm

Communication occurs between the PEs with too few or too many rows of the matrix.

- Communication cost in distribution is low
- Order of the rows changes after distribution
Distribution method: metis dist

1. ParMETIS partitions the graph
2. Numbering in each partitioned domain
3. Matrix distribution

Matrix distribution based on ParMETIS

- The number of edges between domains is expected to be minimized.
- The calculation cost of both ParMETIS and generation of the graph for ParMETIS is needed.
Distribution method 3: constant order

Mat A

- The order of the matrix rows is fixed, but the width of the rows on each PE changes.
Numerical Test

Objectives:

- Are the ordering and the distribution of problem matrix the key elements for performance?
- What happened on performance, if the sparse matrix is distributed repeatedly?
- How different are the performance behaviors on various matrix distribution methods?
Numerical Test

- Poisson problem on Cubic domain
  - 100x100x100:

- Environment:
  - 16 nodes (Xeon 2.8GHz x2 memory1GB) connected by 1000Base
  - LAM/MPI and Intel Fortran Compiler are used

\[
\frac{\partial}{\partial x}(\frac{\partial p}{\partial x}) + \frac{\partial}{\partial y}(\frac{\partial p}{\partial y}) + \frac{\partial}{\partial z}(\frac{\partial p}{\partial z}) = 0
\]

\[
\begin{aligned}
\text{Xmin : } & \frac{\partial p}{\partial x} = 10.0 \\
\text{Ymin : } & \frac{\partial p}{\partial y} = 5.0 \\
\text{Zmax : } & \frac{\partial p}{\partial z} = 1.0 \\
\text{Zmin : } & p = 0.0
\end{aligned}
\]
Numerical Test 1

Initially, the problem matrix is distributed at first 4PEs. Each PE has 250000 rows. After matrix re-distribution, ICCG solver is called.

no redist: matrix distribution isn’t done.
Numerical Test 2

Initially, the problem matrix is distributed at first 4PEs. After 5-time matrix distribution with different weights, ICCG solver is called.
Evaluation for distribution methods

- **lowest comm**
  - The cost of the distribution is low
  - Repeated matrix distribution results in the degradation of the ordering

- **metis dist**
  - The cost of the distribution is high
  - Communication tables are expected to be minimized

- **constant order**
  - The cost of the distribution is low
  - The degradation of the ordering is small even after the repeated distribution
Summary and Conclusion

- In the case that repeated matrix distribution is needed, the matrix distribution method should be considered carefully.
  - The degradation of the ordering often occurs.
  - The cost of distribution may influence the overall performance.
- Various matrix distribution routines can be made using this library
  - In order to create new matrix distribution method, the user have to create only one subroutine which specifies the ordering and the distribution.
Future Work

- Various reordering routines and matrix distribution routines will be created.

- Other sparse matrix data structure than distributed CRS format will be added to the library.