Product Overview

Parallel Direct Sparse Solver for Clusters is a powerful tool set for solving system of linear equations with sparse matrix of millions rows/columns size. Cluster sparse solvers provides an advanced implementation of the modern algorithms and is considered as expansion of Intel MKL Pardiso on cluster computations. For more experienced users, Cluster sparse solvers offers insight into the solvers sufficient to finer tune them for better performance. Cluster sparse solvers is available starting Intel MKL 11.2 Beta

The main features in Cluster sparse solvers functionality:

- Distributed csr format, support distributed matrix, rhs and/or distributed solution
- Solving of system with multiple right hand side
- Cluster support of factorization and solving steps
- C and Fortran examples

A hybrid implementation combines Message Passing Interface (MPI) technology for data exchange between parallel tasks (processes) running on different nodes, and OpenMP* technology for parallelism inside each node of the cluster. This approach effectively uses modern hardware resources such as clusters consisting of nodes with multi-core processors. The solver code is optimized for the latest Intel processors and also performs well on clusters consisting of non-Intel processors.

Cluster sparse solvers provides a Fortran interface, but can be called from C programs by observing Fortran parameter passing and naming conventions used by the supported compilers and operating systems. Code examples are available in the Intel MKL installation examples directory.

Technical Requirements

Please see the Release Notes for more details on technical requirements, including the list of supported processors and operating systems.

Frequently Asked Questions

Q - How do I get started using Parallel Direct Sparse Solvers for Clusters?

A – The first thing that we recommend you to do before starting to use this tool is to make sure that you have reviewed the Reference Manual for Intel® Math Kernel Library and Startup Guide. These guides provide a detailed explanation on how to get started with Cluster sparse solvers. Please pay attention to the Cluster Sparse Solvers requirements, including the availability of the Intel® C++ and Fortran Compilers, as well as the Intel® Math Kernel Library (Intel® MKL).

Q - Where can I get support for the use of this utility?

A - We encourage you to visit our support forum.

Q – How can I switch from Intel MKL Pardiso to cluster version?
A – The main changes described by following algorithm:

Fortran interface:

PARDISO:

Call PARDISO(pt, maxfct, mnum, mtype, phase, n, a, ia, ja, idum, nrhs, iparm, msglvl, b, x, error);

CPARDISO:

Call CPARDISO(pt, maxfct, mnum, mtype, phase, n, a, ia, ja, idum, nrhs, iparm, msglvl, b, x, &error, comm);

C interface:

PARDISO:

PARDISO (pt, &maxfct, &mnum, &mtype, &phase, &n, a, ia, ja, &idum, &nrhs, iparm, &msglvl, b, x, &error);

CPARDISO:

    comm = MPI_Comm_c2f(MPI_COMM_WORLD);

    CPARDISO (pt, &maxfct, &mnum, &mtype, &phase, &n, a, ia, ja, &idum, &nrhs, iparm, &msglvl, b, x, &error, &comm);

Q – How can I put my matrix to Cluster Sparse Solvers interface?

A – There are 2 variants of input data, Column Sparse Row (CSR) that is supported by PARDISO and Distributed CSR (DCSR) that can handle with distributed matrix between processes. The DCSR format is useful for FEM (Finite Element Method) based algorithms.

In a more general FEA (Finite Element Analysis) approach, let's consider a 2D basic mesh topology, formed of 4 elements sharing a central node.
Considering only one degree of freedom per node, the connectivity of such system could be expressed by the following 9x9 symmetric matrix:

$$A = \begin{bmatrix}
1 & * & * & * & * & 1 & 1 \\
1 & 1 & * & * & * & 1 & 1 \\
1 & 1 & * & * & * & * & 1 \\
1 & 1 & * & * & * & * & 1 \\
1 & 1 & * & * & * & * & 1 \\
1 & 1 & * & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 
\end{bmatrix}$$

As mentioned above, in a distributed assembled matrix input format, the above input matrix A should be divided into sequential row subsets, or domains. Each domain belonging to its own MPI process with potentially intersecting neighboring domains. Again for such intersections, the element values of the resulting matrix are calculated as a sum of respective elements of two domains.

Let's consider each element as a specific domain:

The resulting values, columns and rowIndex arrays on each MPI process with their respective distributed connectivity matrix would be:
As mentioned above, each domain is a sequential row subset, delimited by parameter iparm(41) and iparm(42). In the above example, for Domain 1, the local connectivity matrix had to include the missing degrees of freedom (rows from 3 to 7). This can be easily overcome as the compress row-by-row storage restriction for Parallel Direct Sparse Solver for Clusters was alleviated in order to allow zero diagonal element to be omitted.
**Domain 3**

\[
\begin{bmatrix}
0.5 & 1 & 1 & * & * & 0.5 \\
1 & 1 & * & * & 1 \\
0.5 & * & * & 0.5 \\
* & * & * & 0.5 \\
* & * & * & 0.25 \\
\end{bmatrix}, \text{ rows from 4 to 9}
\]

<table>
<thead>
<tr>
<th>Values</th>
<th>0.5</th>
<th>1</th>
<th>1</th>
<th>0.5</th>
<th>1</th>
<th>1</th>
<th>0.5</th>
<th>0.5</th>
<th>0.25</th>
</tr>
</thead>
<tbody>
<tr>
<td>Columns</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>9</td>
<td>5</td>
<td>6</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>RowIndex</td>
<td>1</td>
<td>5</td>
<td>8</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>11</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Domain 4**

\[
\begin{bmatrix}
0.5 & 1 & 1 & 0.5 \\
1 & 1 & 1 & 0.5 \\
0.5 & 0.5 & 0.25 \\
\end{bmatrix}, \text{ rows from 6 to 9}
\]

<table>
<thead>
<tr>
<th>Values</th>
<th>0.5</th>
<th>1</th>
<th>1</th>
<th>0.5</th>
<th>1</th>
<th>1</th>
<th>0.5</th>
<th>0.5</th>
<th>0.25</th>
</tr>
</thead>
<tbody>
<tr>
<td>Columns</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>RowIndex</td>
<td>1</td>
<td>5</td>
<td>8</td>
<td>10</td>
<td>10</td>
<td>11</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Q – Where can I find any papers/performance results of Cluster Sparse Solvers functionality?

A –

- Kalinkin, Anders, Anders “Intel direct sparse solver for clusters, a research project for solving large sparse systems of linear algebraic equations on clusters”, SparseDays-13, Toulouse, Franse
- Kalinkin, Anders, Anders “Intel Direct Sparse Solver for Clusters, a research project for solving large sparse systems of linear algebraic equations on clusters”, Differential Equations. Function Spaces. Approximation Theory conference, August, Novosibirsk, Russia