Domain Cloning – A new approach to parallelization of particle-in-cell (PIC) codes

R. Hatzky and A. Bottino

1. Introduction
2. Structure of a PIC code
3. Optimization on RISC processors
4. Particle counting sort
5. Performance on an IBM 'Regatta' node
6. Decomposition techniques
7. Parallelization of the field solver
8. Summary

Introduction I

Particle-in-cell (PIC) codes are widely used for modeling many body systems, as e.g. in

- **plasma simulations**
- **semiconductor device simulations**
- **gravitational N-body problems**

The number of physical particles N_S in a plasma physics simulation is usually too large to be simulated directly.

A PIC code simulates the motion of N macro particles representing each many physical particles ($N \ll N_S$).

Especially nonlinear simulations are computationally very expensive with $N \approx 10^8$ macro particles and a restrictive time step.

Introduction II

Example: the plasma physics PIC code TORB[1] to simulate the time evolution of ion-temperature-gradient-driven (ITG) turbulence in cylindrical geometry:

- **The numerical method is a particle-mesh method.**
- **The macro particles sample the five-dimensional reduced phase space.**
- **A mesh is needed to calculate the three-dimensional spatial electrostatic potential.**

Minimum number of macro particles: $134 \cdot 10^6$

The smallest mesh size (r, θ, z)**:** $67 \times 259 \times 35$

[1] **R. Hatzky, et al.,** *Phys. Plas.*, **9, 898 (2002).**

Structure of a PIC Code I

For each time step the electrostatic PIC code executes three main parts:[1]

- **Charge deposit (accumulation)** $\approx 40\ \%$
- **Field solve** $< 5\ \%$
- **Electric field calculation** $\approx 45\ \%$
  + time integration (push) $\approx 5\ \%$

The field solve is done by a direct solver (IBM WSMP) with a Cholesky decomposition done at the initialization.

The macro particle communication for a typical 32 **processor run takes** $\approx 5\ \%$**.**

A typical counting sort of the particles takes $\approx 5\ \%$**.**

[1] **V. K. Decyk,** *Comp. Phys. Comm.*, **87, 87 (1995).**

Structure of a PIC Code II

Charge deposition:

1. **The charge of a particle is deposited to nearby grid points.**
2. **Many particles contribute to a certain grid point.**

Example: a particle with the shape of a hat function

```
REAL, DIMENSION ::  q(nx)
n = INT(x)
dx = x - REAL(n)
q(n) = q(n) + 1.0-dx
q(n+1) = q(n+1) + dx
```

Scatter operation:

An indirect memory addressing based on n is used to accumulate the charge in the array q.

A cubic B-spline shape function in three dimensions: one particle contributes to 64 grid points.

Structure of a PIC Code III

Electric field calculation:

1. **The electric field at the particle position is interpolated from nearby grid points.**
2. **Each grid point is read by many particles.**

Example: a particle with the shape of a hat function

```
REAL, DIMENSION ::  Ex(nx)
n = INT(x)
dx = x - REAL(n)
Ep(n) = Ex(n) * (1.0-dx) + Ex(n+1) * dx
```

Gather operation:

An indirect memory addressing in the array `Ex` is used to interpolate the electric field at the particle position.

A cubic B-spline interpolation in three dimensions: 64 grid points sum up to the electric field.

Optimization on RISC Processors

General optimization strategies:[1]

- **Storing particle components together in one array.**
- **Sorting particle and field components together.**
- **Removing `IF` statements by using guard cells.**

Cache based RISC architecture:

Cache trashing occurs when the charge and/or electric field array do not fit into the cache.

A maximum cache reuse is given when all the particles in the same mesh cell are processed together.

Solution:

Counting sort of the particles into the bins of the grid.

[1] **V. K. Decyk, et al.,** *Comp. Phys.*, **10, 290 (1996).**

Particle counting sort[1] O(N)

Count the number of particles in each cell

```
DO n = 1, N
   i = compute the cell for particle I_n
   P_i = P_i + 1
END DO
```

Convert P into an allocation

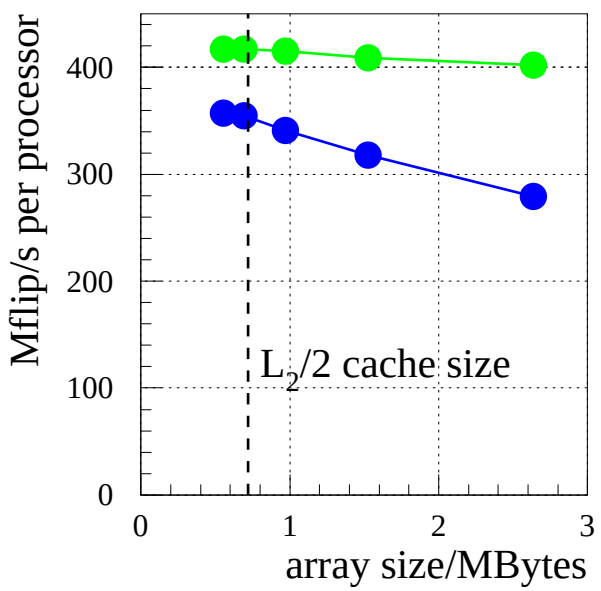
```
DO n = 1, M
   j = P_i
   P_i = k
   k = k + j
END DO
```

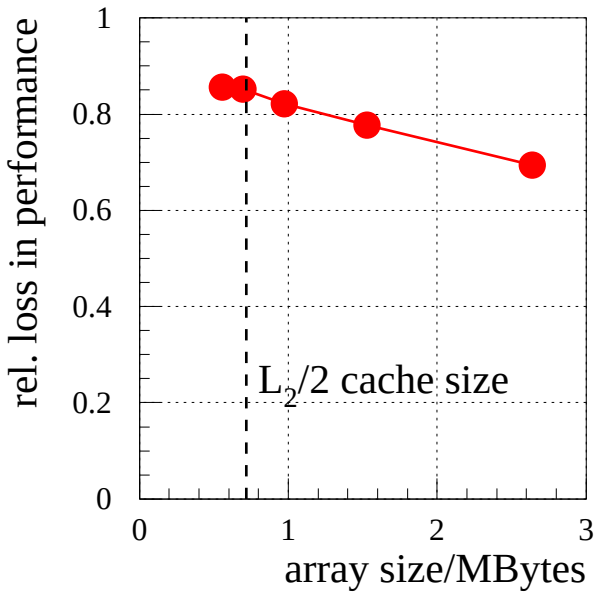
Sort the particles I into O

```
DO n = 1, N
   i = compute the cell for particle I_n
   j = P_i
   P_i = P_i + 1
   O_j = I_n
END DO
```

[1] **K. J. Bowers,** *J. Comp. Phys.*, **173, 393 (2001).**

Particle counting sort II

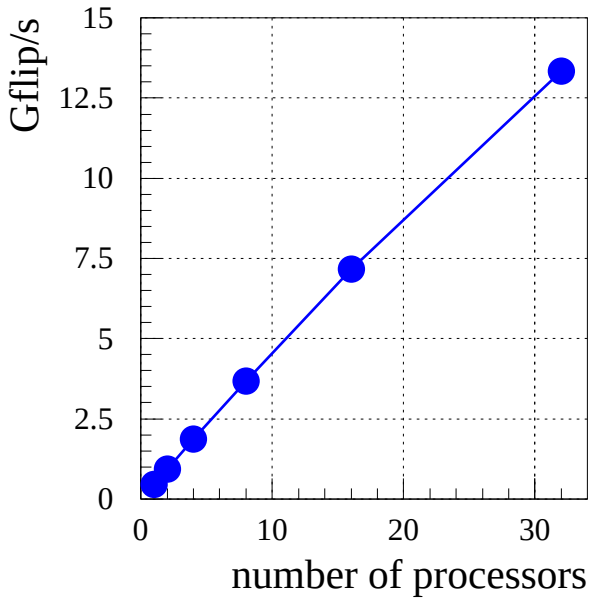


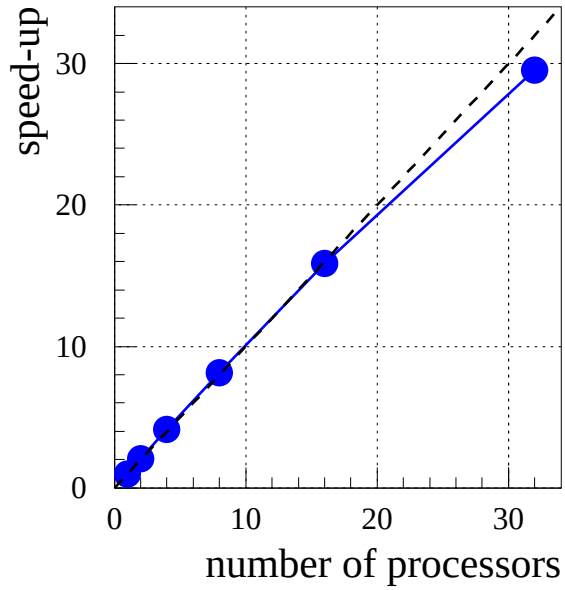


Degradation of the performance with larger array size due to the cache architecture of the IBM Power4 processor:

- **Degradation starts at** $\approx L_2/2 = 0.72$ **MBytes**
- **Only slight degradation with particle sort**
- **Rel. performance loss without particle sort** $> 30\ \%$
- **Already gain of** $\approx 15\ \%$ **for small arrays**

Performance on an IBM 'Regatta' node





Communication of the particles among the processors (IBM Power4) only in large packages:

- **Nearly perfect scaling for ≤ 16 processors**
- **Overall speed-up of ≈ 29**

$\Rightarrow$ **good scalability with the number of processors.**

But only $\approx 12\ \%$ of the peak performance.

Decomposition techniques I

Domain decomposition:

Different portions of the physical domain are assigned to different processors, including the particles and field data that reside on them.

Advantage:

- **The field data are local**
  $\Rightarrow$ **small memory consumption.**

Disadvantages:

- **The particles have to be communicated among the subdomains (processors).**
- **The number of neighboring subdomains scales with the number of processors:**
  - $\Rightarrow$ **the spatial extent in the direction of parallelization becomes smaller and smaller.**
  - $\Rightarrow$ **the particle communication is increased; the minimal grid resolution is increased.**

Decomposition techniques II

Particle decomposition:

The whole spatial grid is assigned to each processor, which takes care only of a subset of the particles.

Advantages:

- **Intrinsic load balancing**
- **No particle communication**
- **Moderate effort in porting existing sequential programs into parallel form**

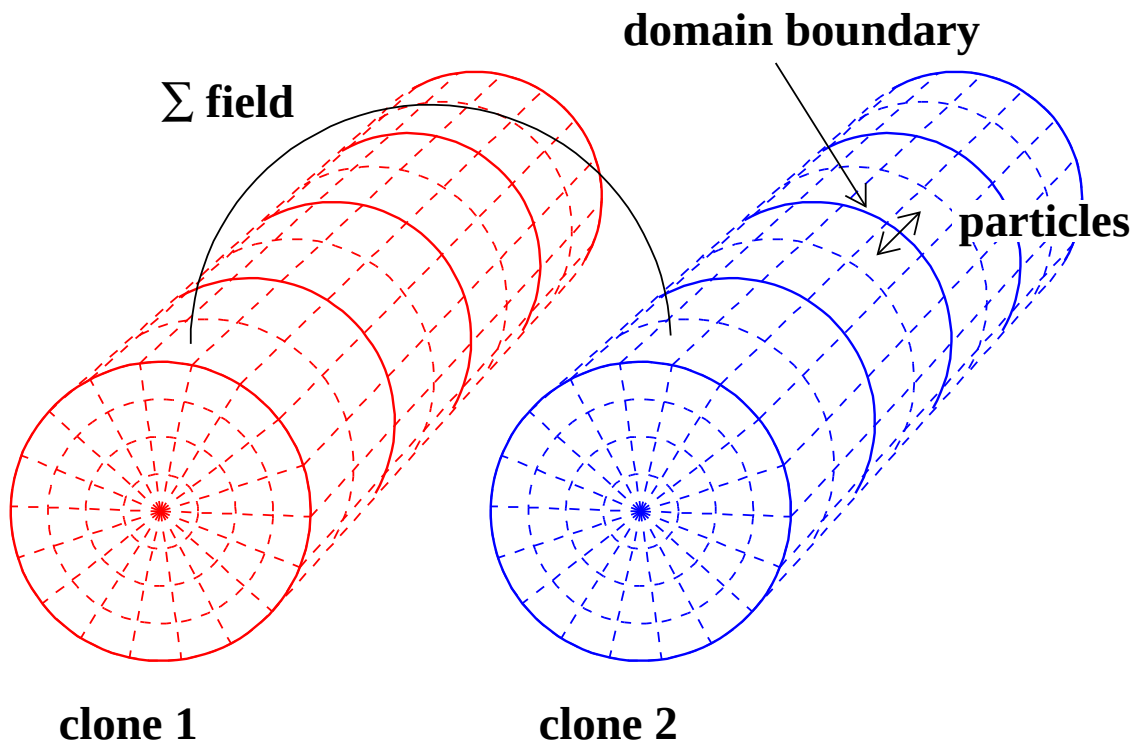
Disadvantages:

- **A global reduction communication for the field data.**
- **The field data have to be stored on each processor $\Rightarrow$ high memory consumption.**
- **A large average number of particles per grid cell on the single processor is needed (cache reuse).**

Decomposition techniques III

Domain cloning:[1]

The simulation domain is cloned, i.e. multiple copies of the same domain are made.



[1] **C. C. Kim and S. E. Parker,** *J. Comp. Phys.*, **161, 589 (2000).**

Decomposition techniques IV

Particle MPI communicator comm_push

clone 1	clone 2
me_push=0	me_push=0
me_push=1	me_push=1
me_push=2	me_push=2
me_push=3	me_push=3

Charge-assignment MPI communicator comm_assign

clone 1	clone 2
me_assign=0	me_assign=1
me_assign=0	me_assign=1
me_assign=0	me_assign=1
me_assign=0	me_assign=1

Decomposition techniques V

General advantages:

- Domain decomposition and particle decomposition are included as limiting cases.
- No correlation between grid resolution and number of processors.
- The implementation can be easily performed as a supplement to one-dimensional domain decomposition.

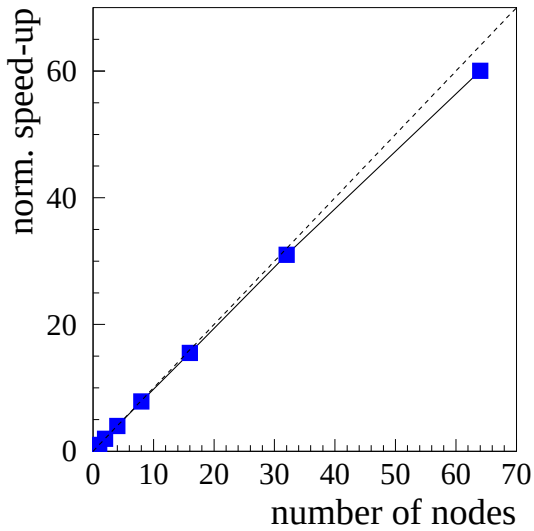
Possible advantages on a clustered SMP computer:[1]

- Dominating particle communication inside a node $\Rightarrow$ occurs on the fast shared memory.
- Moderate field communication between the nodes $\Rightarrow$ passes the slower network (IBM Federation switch).

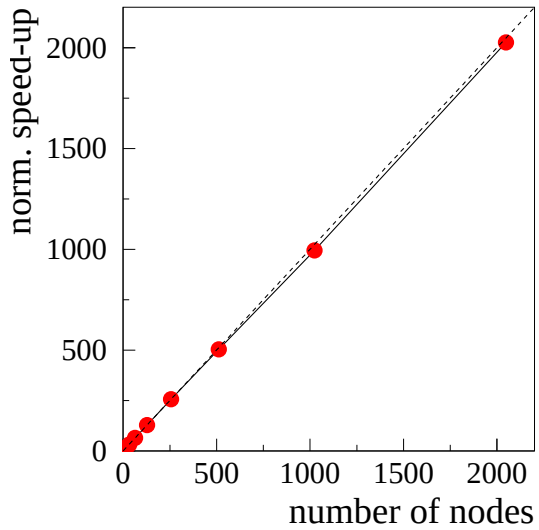
[1] R. Hatzky, *Parallel Computing*, 32, 325, 2006.

Decomposition techniques VI

ECMWF run on IBM pSeries 690+ servers (node) each of which has 32 1.9 GHz Power 4+ CPUs:



BSC run on 2.2 GHz IBM Power PC 970FX dual blade nodes:



Field solver I

3D matrix is transformed by an FFT decomposition in toroidal direction into a system of 2D matrix equations.

Advantage:

- **Trivial parallelization on each clone over the Fourier modes.**

Disadvantage:

- **2 FFT's per field solving** $[\mathcal{O}(N \log_2 N)]$

Using the MPI communicator comm_assign for parallelization of the field solving.

Advantage:

- **Parallelization over the clones avoids redundant field solving.**

Disadvantage:

- **The factorized and parallelized matrix has to be stored several times $\Rightarrow$ conflicts memory limitation.**

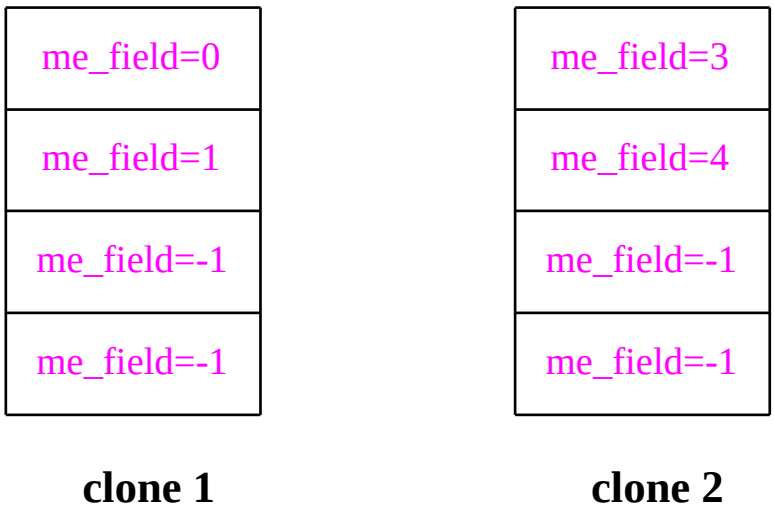
Field solver II

Especially for large matrices a third MPI communicator comm_field has to be defined which is completely independent from the domain cloning concept.

Advantage:

- **Maximal flexibility to adapt to the scaling properties and memory consumption of the chosen parallel solver.**

Field solver MPI communicator comm_field



Field solver III

The ORB5[1] code solves simultaneously:

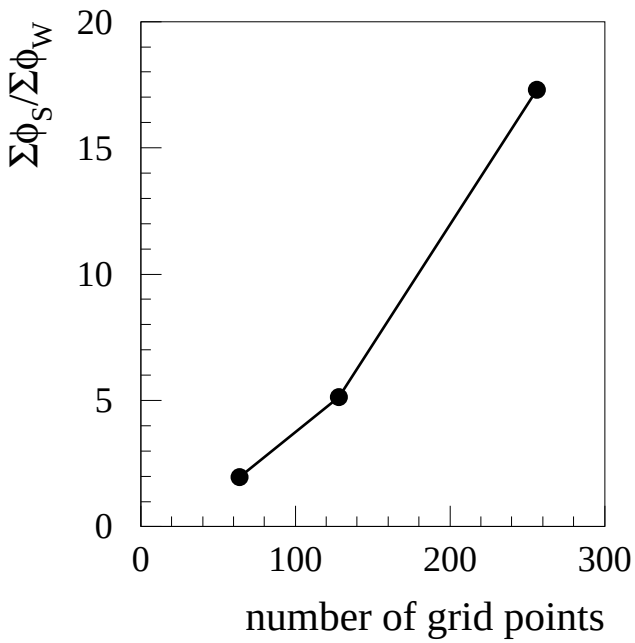
1. **a sparse Helmholtz equation for ϕ using the MPI communicator comm_assign**
2. **a dense integro-differential equation for $\bar{\phi}$ using the MPI communicator comm_field**

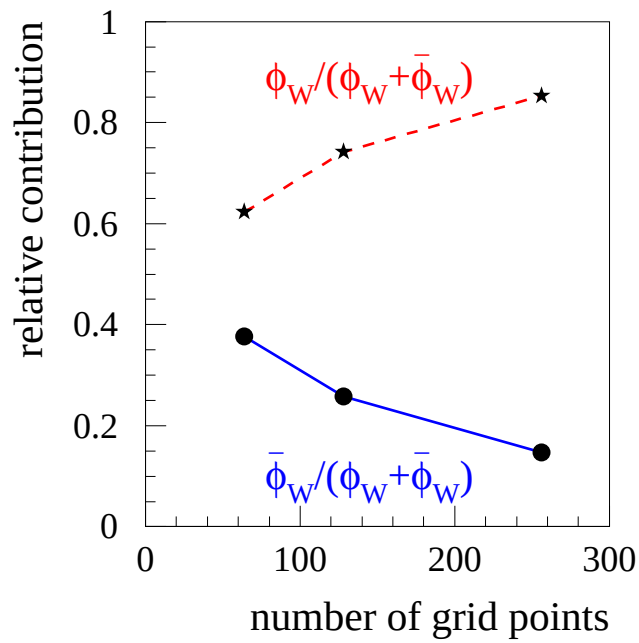
Benchmarks with two direct solvers have been done:

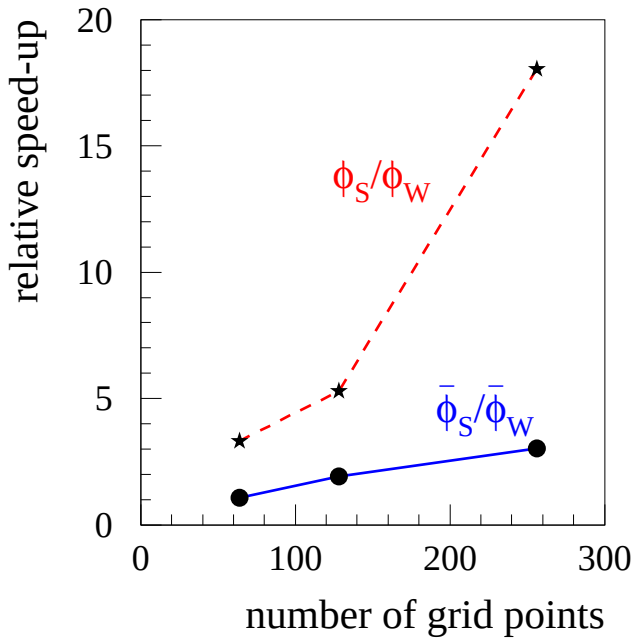
1. **the publicly available library ScaLAPACK for dense matrices**
2. **the commercial IBM library WSMP for sparse matrices**

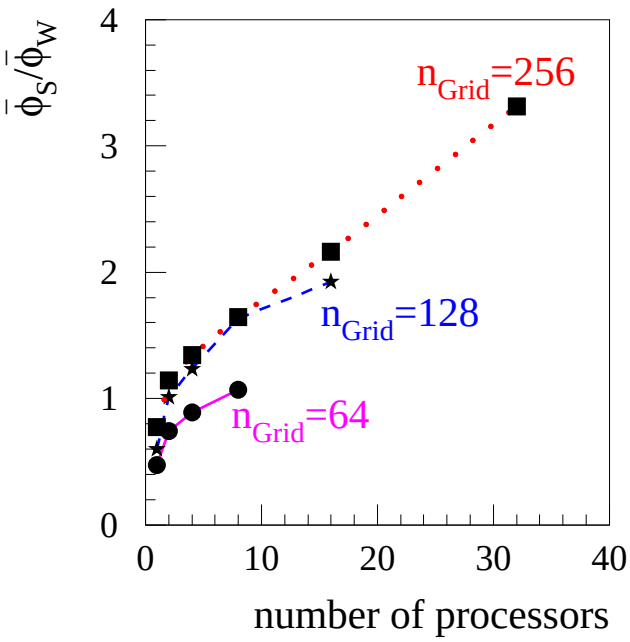
[1] **P. Angelino, et al.,** *Plasma Physics and Controlled Fusion,* **48: 557 (2006).**

Benchmark ScaLAPACK vs. IBM WSMP









Summary

- **A particle counting sort enhances the performance on the POWER4 microarchitecture due to a better cache reuse.**

  Especially for large charge and/or electric field arrays, which do not fit into the L_2 cache, a performance gain of $> 30\ \%$ is possible.

- **The domain cloning decomposition concept decouples the grid resolution from the number of processors used.**

  Nearly optimal scaling properties up to 2048 (ECMWF) and 4096 (BSC) processors could be achieved for the `TORB` code surpassing the Tflop/s threshold.

- **The IBM WSMP library seems to be the optimal choice to solve for large sparse matrices.**