# Implementing SuperLU on Contact Problems

**Kengo Nakajima**
**RIST, FSIS/University of Tokyo and CREST/JST**

**4th ACTS Workshop**
**August 2003.**

# Why SuperLU ?

- One Alternative for Ill-Conditioned Problems
  - Contact
  - MPC (Multiple Point Constraint)
  - Shell Elements

# Current Status ...

- Just Started
- Only Single PE Version
- Expensive ...
  - Memory: 9x memory for connectivity
  - CPU
- Optimized BLAS Library ?

- I expect that I can improve efficiency and try parallel cases during this workshop !! Moreover I would like to try other libraries in ACTS.

# Contact Problems

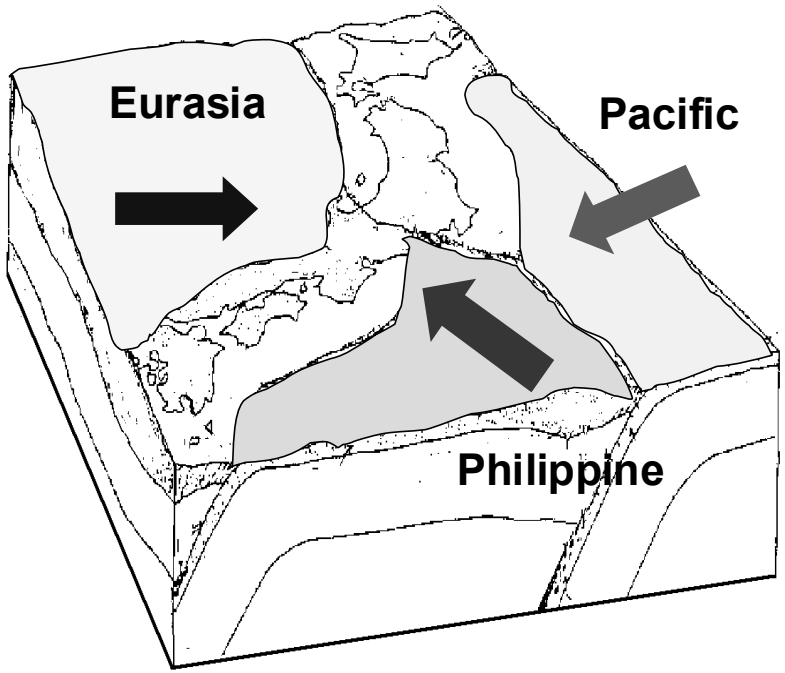
- Applications in geophysical science
  - Earthquake Generation Cycle
- Ill-conditioned problems due to penalty constraint for fault-zone contact.
- Selective-blocking (SB) preconditioning developed by Kengo Nakajima provides robust convergence efficiently ... but very limited use for parallel computation.
  - ALL nodes in contact group should be on same partition !!

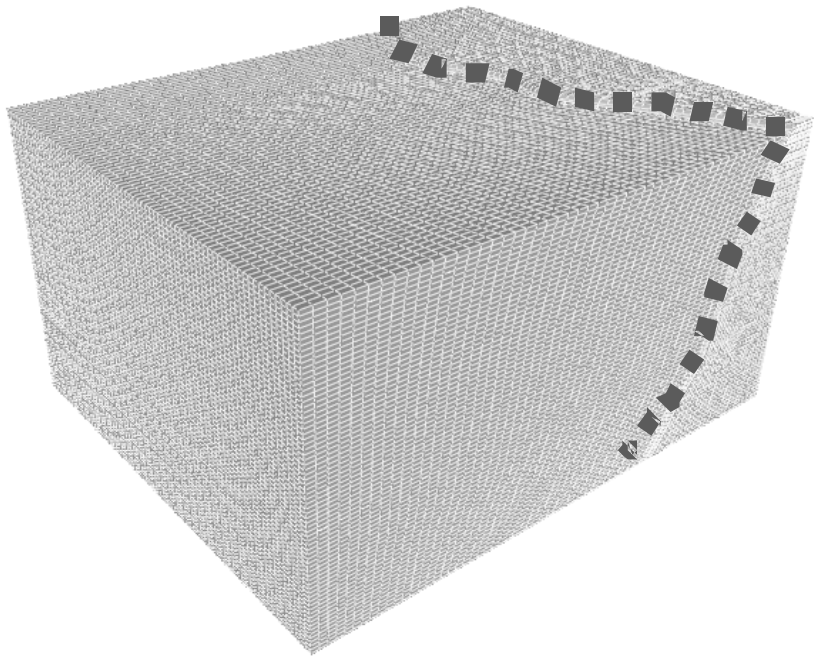
# Contact Problems (cont.)

- SAI by C.Shen & J.Zhang (Univ. Kentucky)
  - Excellent robustness even if nodes in contact groups are *decoupled*.
  - Set-up time is not scalable in parallel computing.
  - Number of Iterations for convergence stays constant for multiple domains.

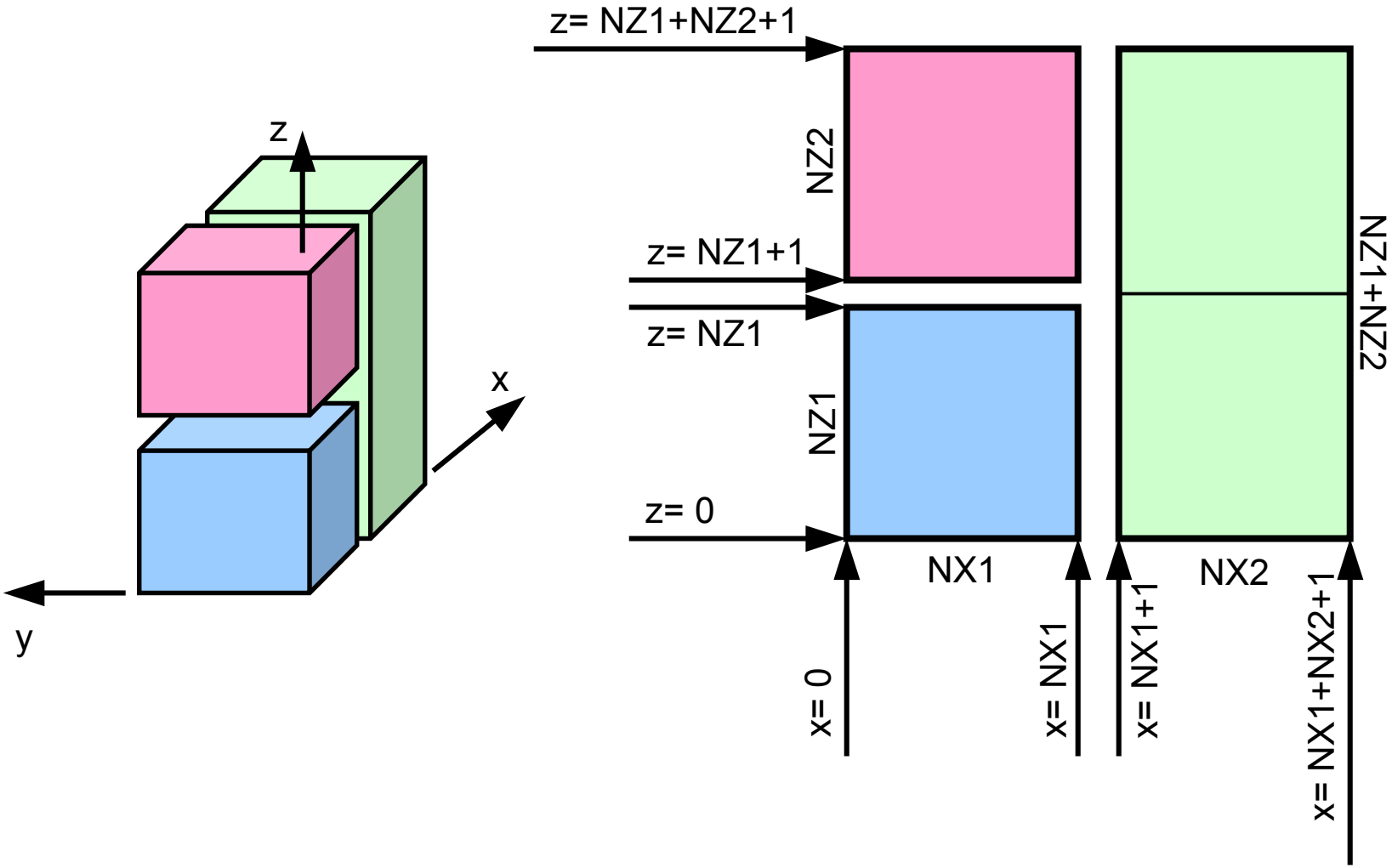
# Geophysics Appl. w/Contact

Augmented Lagrangean Method with Penalty Constraint Condition for Contact

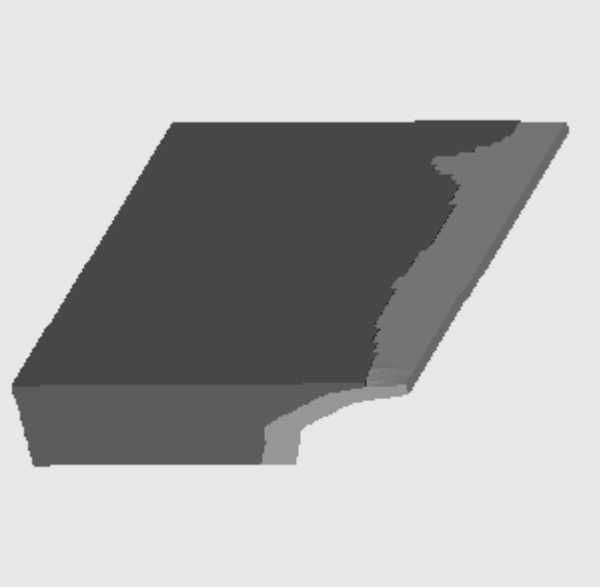


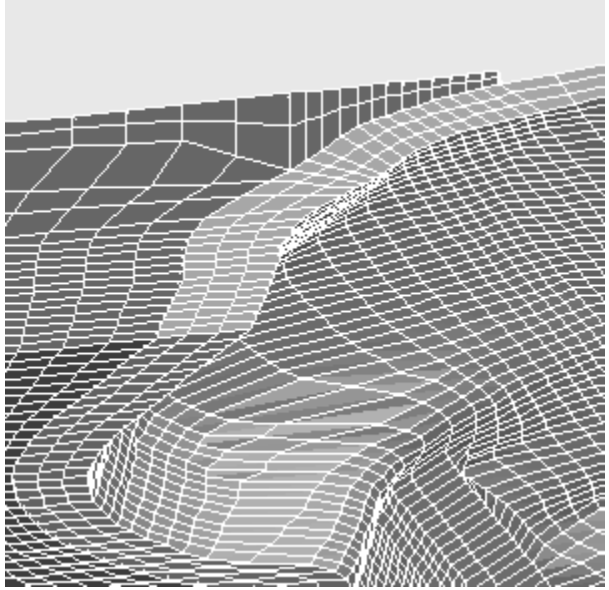



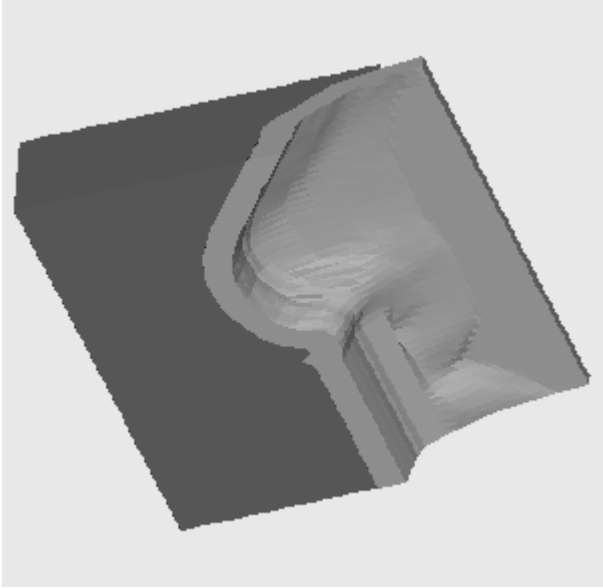
# Simple Block Model



# SWJ: SouthWest Japan



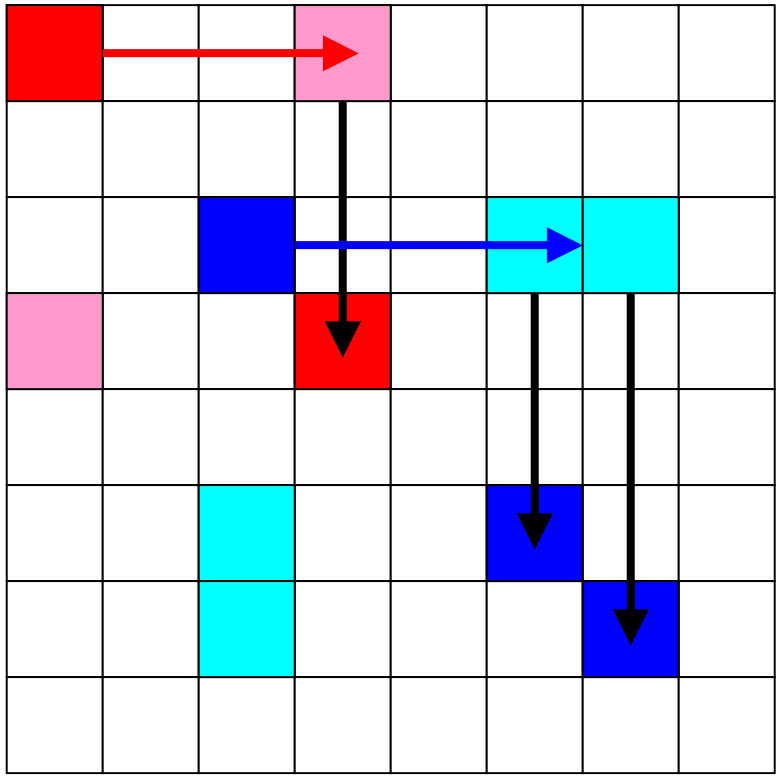




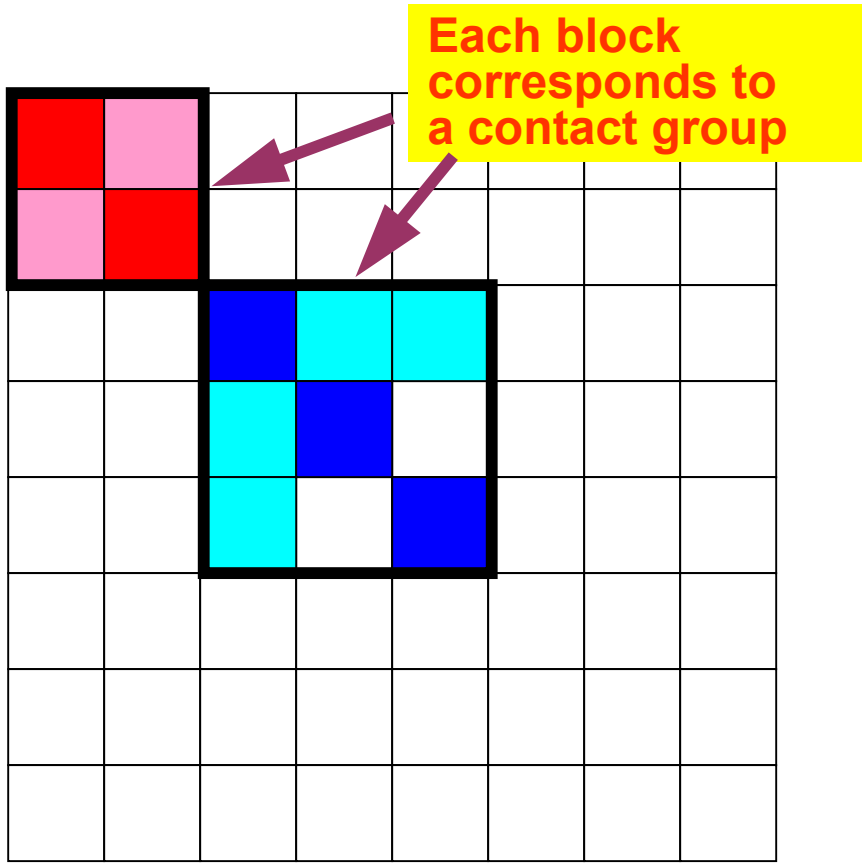
# Selective Blocking

## Special Method for Contact Problem

Strongly coupled nodes are put into the same diagonal block.



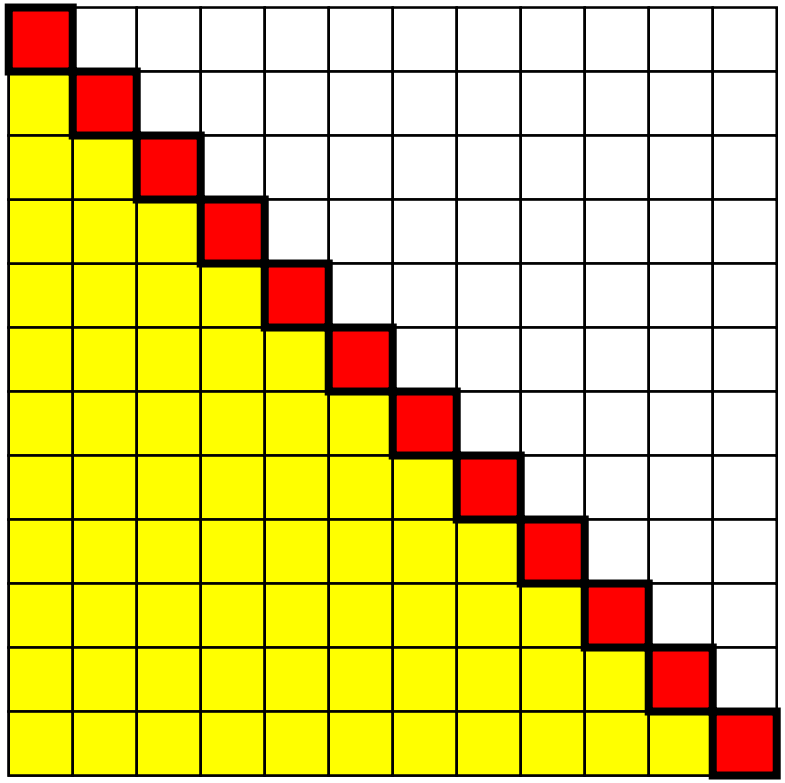
Initial Coef. Matrix
find strongly coupled contact groups (each small square:3x3)



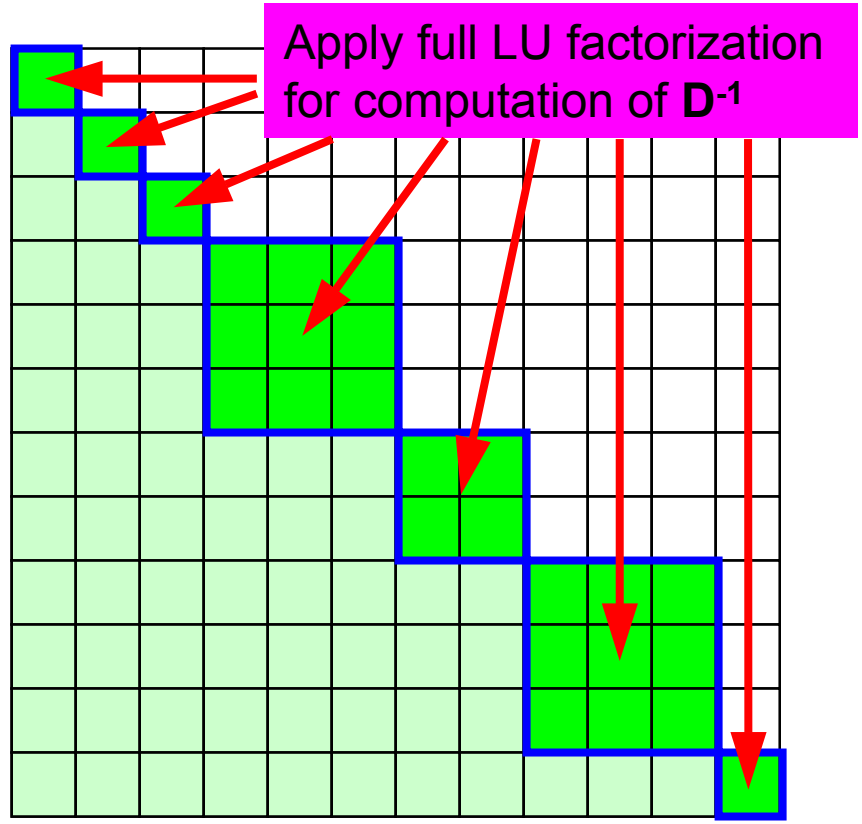

Reordered/Blocked Matrix
nodes/block

# Selective Blocking or Supernode

Procedure : Forward Substitution in Lower Tri. Part



**Block ILU/IC**



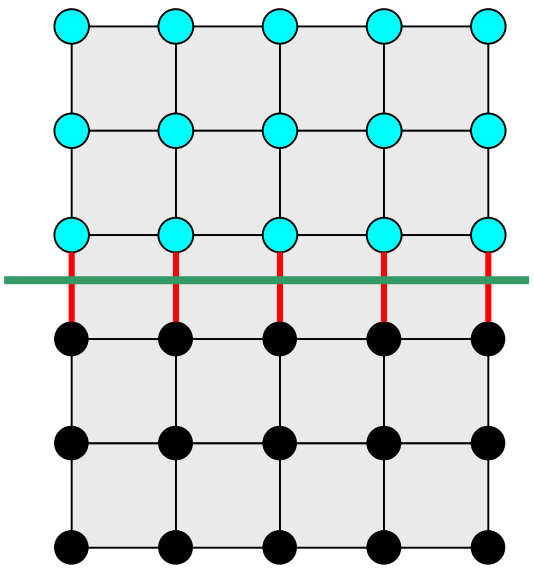

**Selective Blocking/ Supernode**

size of each diagonal block depends on contact group size

# Outline of the Repartitioning

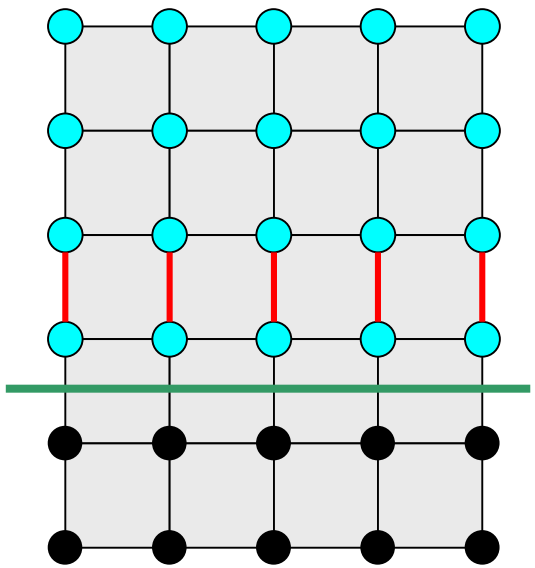
Convergence is slow if nodes in each contact group locate on different partition.

Repartitioning so that nodes in contact pairs would be in same partition as INTERIOR nodes will be effective.



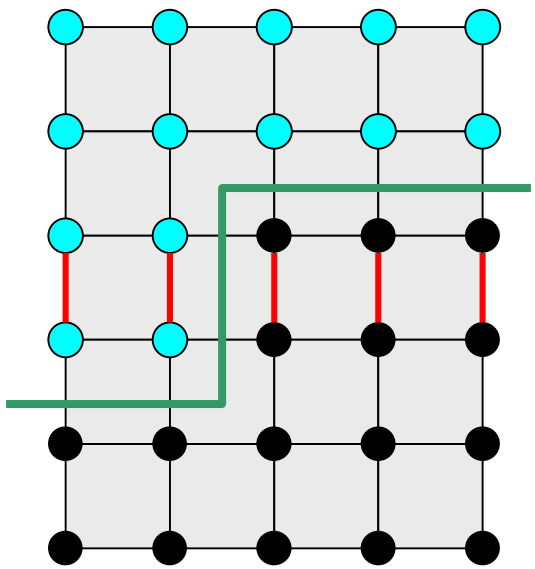
**BEFORE repartitioning**

**Nodes in contact pairs are on separated partition.**



**AFTER repartitioning**

**Nodes in contact pairs are on same partition, but no load-balancing.**



**AFTER load-balancing**

**Nodes in contact pairs are on same partition, and load-balanced.**

# Special Repartitioning

## Benchmark: 4 PE cases

### BEFORE Repartitioning

| Precond. | $\lambda$ | Iter # | sec. |
|---|---|---|---|
| BIC(1) | $10^{10}$ | 90 | 4.1 |
| | $10^{16}$ | 1,724 | 70.7 |
| BIC(2) | $10^{10}$ | 86 | 6.6 |
| | $10^{16}$ | 962 | 59.8 |
| | $10^{20}$ | No Conv. | N/A |
| SB-BIC(0) | $10^{10}$ | 156 | 3.5 |
| | $10^{16}$ | 1,598 | 33.9 |
| | $10^{20}$ | 2,345 | 55.5 |

### AFTER Repartitioning

| Precond. | $\lambda$ | Iter # | sec. |
|---|---|---|---|
| BIC(1) | $10^{10}$ | 80 | 3.8 |
| | $10^{16}$ | 167 | 7.4 |
| BIC(2) | $10^{10}$ | 71 | 5.8 |
| | $10^{16}$ | 74 | 5.9 |
| | $10^{20}$ | No Conv. | N/A |
| SB-BIC(0) | $10^{10}$ | 126 | 2.9 |
| | $10^{16}$ | 124 | 2.8 |
| | $10^{20}$ | 231 | 5.7 |

# Results on Hitachi SR2201

## Parallel Performance of SB-BIC(0)-CG

NX1=NX2=70, NY=40, NZ1=NZ2=70, **Repartitioned.**
2,471,439 DOF, 784,000 Elements, **$\lambda/E=10^6$**
Iterations/CPU time until convergence ($\varepsilon=10^{-8}$)

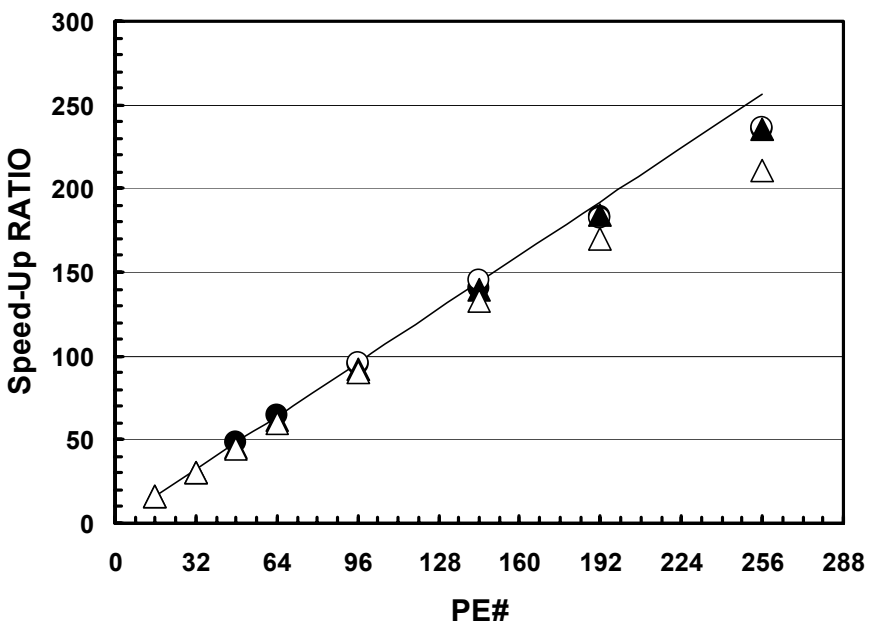
| Preconditioning | | 16 PEs | 32 PEs | 48 PEs | 64 PEs | 96 PEs | 144 PEs | 192 PEs | 256 PEs | Memory Size (GB) |
|---|---|---|---|---|---|---|---|---|---|---|
| BIC(0) | Iterations | 14459 | 14583 | 15018 | 15321 | 15523 | 15820 | 16084 | 16267 | |
| | sec. | 13500 | 7170 | 4810 | 3630 | 2410 | 1630 | 1270 | 1230 | 3.10 |
| | Speed-up | 16 | 30 | 45 | 60 | 90 | 133 | 170 | 211 | |
| BIC(1) | Iterations | | | 379 | 390 | 402 | 424 | 428 | 452 | |
| | sec. | N/A | N/A | 236 | 175 | 119 | 81 | 62 | 48 | 8.39 |
| | Speed-up | | | 48 | 65 | 95 | 140 | 183 | 236 | |
| BIC(2) | Iterations | | | | | 364 | 387 | 398 | 419 | |
| | sec. | N/A | N/A | N/A | N/A | 212 | 140 | 112 | 86 | 14.4 |
| | Speed-up | | | | | 96 | 145 | 182 | 217 | |
| SB-BIC(0) | Iterations | 511 | 524 | 527 | 538 | 543 | 567 | 569 | 584 | |
| | sec. | 555 | 295 | 193 | 144 | 96 | 64 | 48 | 38 | 3.52 |
| | Speed-up | 16 | 30 | 46 | 62 | 92 | 139 | 185 | 235 | |

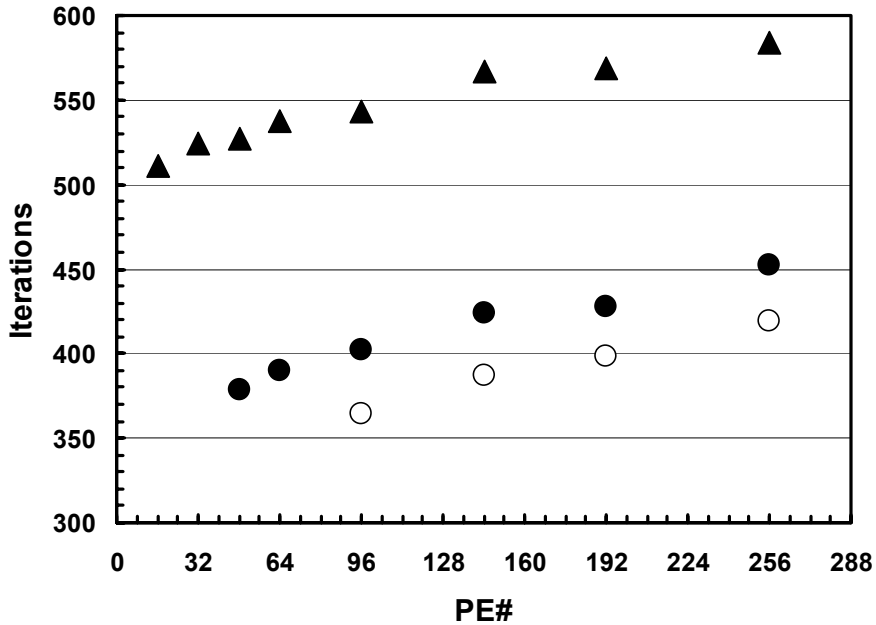
# Results on Hitachi SR2201

## Parallel Performance of SB-BIC(0)-CG

NX1=NX2=70, NY=40, NZ1=NZ2=70, **Repartitioned.**
2,471,439 DOF, 784,000 Elements, **$\lambda/E=10^6$**
Iterations/CPU time until convergence ($\varepsilon=10^{-8}$)

**Black Circles: BIC(1), White Circles: BIC(2)**
**Black Triangles: SB-BIC(0), White Triangles: BIC(0)**

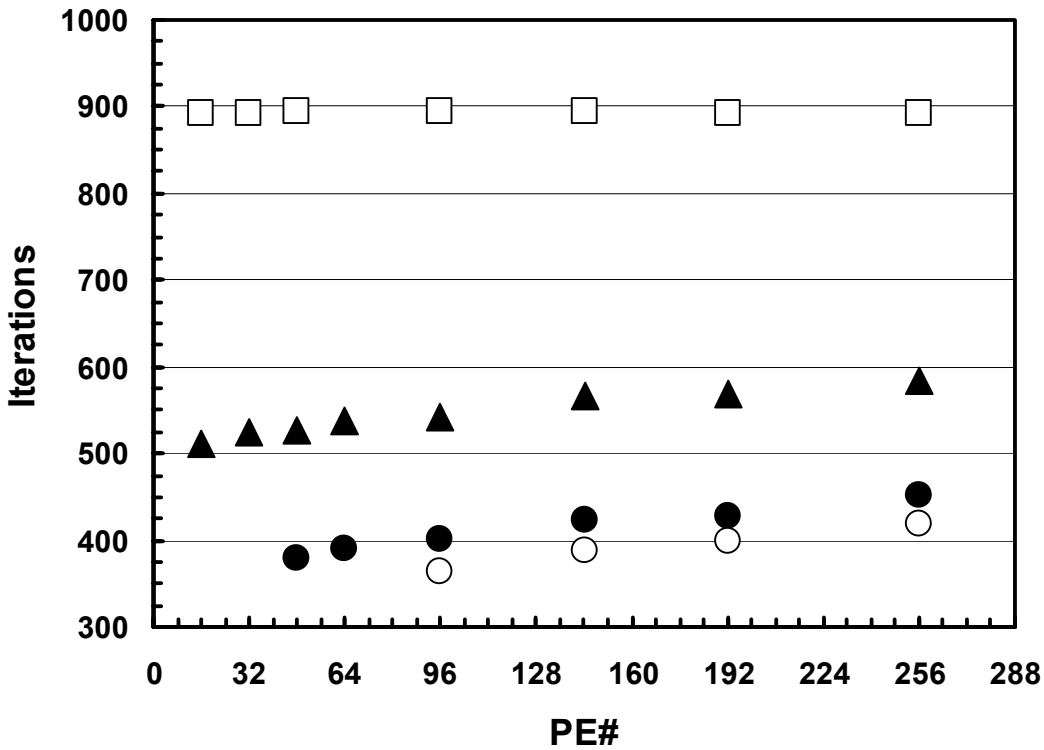


# Results on Hitachi SR2201

## Parallel Performance of SB-BIC(0)-CG

NX1=NX2=70, NY=40, NZ1=NZ2=70, **Repartitioned.**
2,471,439 DOF, 784,000 Elements, **$\lambda/E=10^6$**
Iterations until convergence ($\varepsilon=10^{-8}$)




**Black Circles: BIC(1),**
**White Circles: BIC(2)**
**Black Triangles: SB-BIC(0)**
**White Squares : SAI**

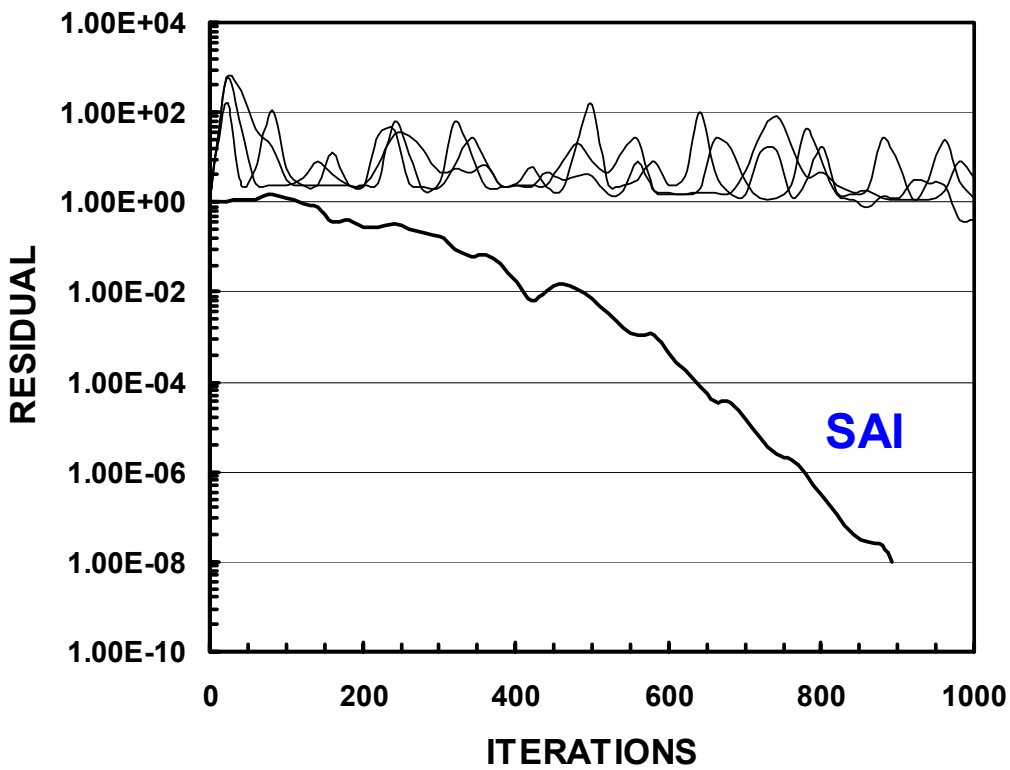
# Results on Hitachi SR2201

## Parallel Performance of SB-BIC(0)-CG

NX1=NX2=70, NY=40, NZ1=NZ2=70, **NOT repartitioned**

2,471,439 DOF, 784,000 Elements, **$\lambda/E=10^6$**

Convergence history for 256 PEs




**BIC(1), BIC(2), SB-BIC(0)**

# Contact Model

## Intel Xeon 2.8 GHz, 1 PE, PGI Compiler
## 47,061 DOF (BLAS library with SuperLU)

| $\lambda$ | BIC(0) (sec./iter's) | BIC(1) (sec./iter's) | BIC(2) (sec./iter's) | SB-BIC(0) (sec./iter's) | SuperLU (sec.) | | |
|---|---|---|---|---|---|---|---|
| | | | | | Fact. | Solve | Total |
| $10^2$ | 17.88<br>365 | 8.51<br>73 | 13.05<br>57 | 6.22<br>108 | 151.81 | 0.45 | 152.89 |
| $10^4$ | 62.55<br>1,271 | 8.47<br>73 | 13.00<br>57 | 6.22<br>108 | 152.00 | 0.44 | 153.09 |
| $10^6$ | 106.96<br>2,201 | 8.50<br>73 | 12.98<br>57 | 6.22<br>108 | 151.27 | 0.44 | 153.35 |

# Simple Cube Model (no contact)

HPC-MW/Scalar BIC(0)-CG vs SuperLU (approx MDO)

Intel Xeon 2.8 GHz, 1 PE, PGI Compiler (BLAS library with SuperLU)

| nodes | DOF | BIC(0)-CG (sec./iter's) | SuperLU (sec.) | | |
|---|---|---|---|---|---|
| | | | Factorization | Solve | Total |
| 16×16×16 | 12,288 | 0.95/42 | 16.38 | 0.09 | 17.18 |
| 20×20×20 | 24,000 | 2.36/52 | 67.03 | 0.27 | 69.75 |
| 24×24×24 | 41,472 | 5.06/62 | 270.40 | 0.52 | 271.52 |
| 28×28×28 | 65,856 | 9.42/72 | N/A | N/A | N/A |
| 32×32×32 | 98,304 | 16.39/82 | N/A | N/A | N/A |

# Interface between F90 & C

```
implicit REAL*8 (A-H,O-Z)
integer, dimension(:), allocatable :: rowind, colptr
real(kind=8), dimension(:), allocatable :: values, b
integer :: n, nnz, nrhs, ldb, info, maxn, maxnz

open  (62, file='mat.p', status='unknown', form='unformatted')
  read (62) n, nnz
  allocate (colptr(n+1), rowind(nnz), values(nnz), b(n))
  read (62) (colptr(i), i= 1, n+1)
  read (62) (rowind(k), k= 1, nnz)
  read (62) (values(k), k= 1, nnz)
  read (62) (b     (k), k= 1, n)
close (62)

nrhs= 1
ldb = n

call c_bridge_dgssv( n, nnz, nrhs, values, rowind, colptr, b, ldb, info )

stop
end
```