Left-Preconditioned Communication-Avoiding Conjugate Gradient Methods for Multiphase CFD Simulations on the K Computer

Akie Mayumi[1], Yasuhiro Idomura[1], Takuya Ina[1],
Susumu Yamada[1], Toshiyuki Imamura[2]

[1]Japan Atomic Energy Agency [2]RIKEN

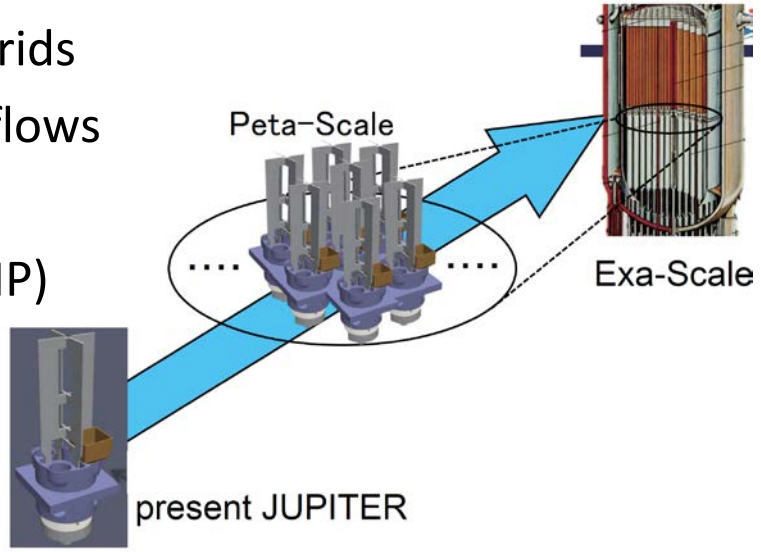
Acknowledgements

- T. Kawamura, S. Yamashita (JAEA)
- This work is supported by the MEXT, Grant for Post-K priority issue No. 6: Development of Innovative Clean Energy, and the computation is performed on the K-computer at the Riken (hp160208).

Exa-scale simulations for severe accident analysis

- JAEA promotes the development of multiphase thermal-hydraulic CFD code for analyzing severe accidents in the Fukushima Daiichi Nuclear Power Plant

- JUPITER code [Yamashita,ICONE2013] simulates relocation of molten materials in nuclear reactors as incompressible viscous fluids.
  - Finite difference method in structured grids
  - Volume of fluid method for multiphase flows
  - Multi-components (UO_2, Zry, B_4C, SUS)
  - 3D domain decomposition (MPI+OpenMP)



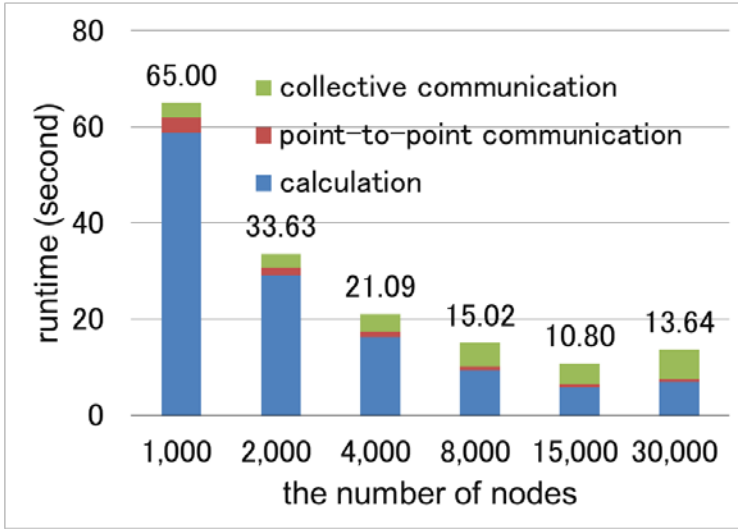
- Target problems
  - Peta-scale (K-computer)
    - Simulate melt-relocation behavior of several fuel assemblies
  - Exa-scale
    - Severe accident analysis for whole reactor pressure vessel

Scalability issue of Poisson solver on K-computer

- K-computer(11.3PF/82,944nodes)
  - SPARC64VIIIfx(128GF=16GF x 8cores, B/F=0.5)
  - Tofu Interconnect(3D torus, 5GB/s x 4ways)
- Poisson solver in JUPITER
  - 2nd order centered finite difference in structured grids (7-stencils)
  - CG method with Block-Jacobi preconditioner (PETSc/in-house solver)
  - Occupy over 90% of computational cost for large problems
  - Large density contrast of multiphase flows gives ill-conditioned problem, and conditions become worse in large problems

  →proper preconditioning is essential
- Scalability issue
  - P2P comm. scales on 3D torus
  - Collective comm. is bottleneck

→In this work, we resolve this issue using Communication Avoiding CG methods



Strong scaling of CG method on K-computer

Communication-Avoiding CG method

Review of CA-CG method based on [Hoemmen,PhD2010]

- CG

dependency between r_j and p_j

Algorithm 1 Conjugate Gradient method

Input: $Ax = b$, Initial guess x_1

Output: Approximate solution x_j

1: $r_1 := b - Ax_1, p_1 := r_1$
2: **for** $j = 1, 2, \ldots$ until convergence **do**
3: Compute $w := Ap_j$
4: $\alpha_j := \langle r_j, r_j \rangle / \langle w, p_j \rangle$
5: $x_{j+1} := x_j + \alpha_j p_j$
6: $r_{j+1} := r_j - \alpha_j w$
7: $\beta_j := \langle r_{j+1}, r_{j+1} \rangle / \langle r_j, r_j \rangle$
8: $p_{j+1} := r_{j+1} + \beta_j p_j$; test convergence
9: **end for**

P2P Comm.

Collective Comm.

Communication-Avoiding CG method

Review of CA-CG method based on [Hoemmen,PhD2010]

- CG

dependency between r_j and p_j

- CG with 3-term recurrence

use only r_j (decouple r_j-p_j dependency)

Algorithm 2 CG3 (3-term recurrence variant of CG)

Input: $Ax = b$, Initial guess x_1

Output: Approximate solution x_j

1: $x_0 := 0, r_0 := 0, r_1 := b - Ax_1$
2: **for** $j = 1, 2, \ldots$ until convergence **do**
3: Compute $w_j := Ar_j$ P2P Comm.
4: $\mu_j := \langle r_j, r_j \rangle$; test for convergence
5: $\nu_j := \langle w_j, r_j \rangle$ Collective Comm.
6: $\gamma_j := \mu_j / \nu_j$
7: **if** $j = 1$ **then**
8: $\rho_j := 1$
9: **else**
10: $\rho_j := (1 - \frac{\gamma_j}{\gamma_{j-1}} \cdot \frac{\mu_j}{\mu_{j-1}} \cdot \frac{1}{\rho_{j-1}})^{-1}$
11: **end if**
12: $x_{j+1} := \rho_j(x_j + \gamma_j r_j) + (1 - \rho_j)x_{j-1}$
13: $r_{j+1} := \rho_j(r_j + \gamma_j w_j) + (1 - \rho_j)r_{j-1}$
14: **end for**

Communication-Avoiding CG method

Review of CA-CG method based on [Hoemmen,PhD2010]

- CG

dependency between r_j and p_j

- CG with 3-term recurrence

use only r_j (decouple r_j-p_j dependency)

- CG with outer/inner loops

Algorithm 3 CG3 with outer and inner loops

Input: $Ax = b$, Initial guess x_1

Output: Approximate solution x_j

1: $x_0 := 0, r_0 := 0, r_1 := b - Ax_1$
2: **for** $k = 0, 1, 2, \ldots$ until convergence **do**
3: **for** $j = 1, 2, \ldots, s$ **do**
4: Compute $w_{sk+j} := Ar_{sk+j}$ P2P Comm.
5: $\mu_{sk+j} := \langle r_{sk+j}, r_{sk+j} \rangle$; test for convergence
6: $\nu_{sk+j} := \langle w_{sk+j}, r_{sk+j} \rangle$ Collective Comm.
7: $\gamma_{sk+j} := \mu_{sk+j} / \nu_{sk+j}$
8: **if** $sk + j = 1$ **then**
9: $\rho_{sk+j} := 1$
10: **else**
11: $\rho_{sk+j} := (1 - \frac{\gamma_{sk+j}}{\gamma_{sk+j-1}} \cdot \frac{\mu_{sk+j}}{\mu_{sk+j-1}} \cdot \frac{1}{\rho_{sk+j-1}})^{-1}$
12: **end if**
13: $x_{sk+j+1} := \rho_{sk+j}(x_{sk+j} + \gamma_{sk+j} r_{sk+j}) + (1 - \rho_{sk+j}) x_{sk+j-1}$
14: $r_{sk+j+1} := \rho_{sk+j}(r_{sk+j} + \gamma_{sk+j} w_{sk+j}) + (1 - \rho_{sk+j}) r_{sk+j-1}$
15: **end for**
16: **end for**

Communication-Avoiding CG method

Review of CA-CG method based on [Hoemmen,PhD2010]

- CG

dependency between r_j and p_j

- CG with 3-term recurrence

use only r_j (decouple r_j-p_j dependency)

- CG with outer/inner loops
- CG with Matrix Power Kernel

compute Ar_{sk+j} based on recurrence formula using basis vectors given by MPK

→CA for halo comm.

Algorithm 4 CA-CG with MPK

Input: $Ax = b$, Initial guess x_1

Output: Approximate solution x_{sk+j}

1: $x_0 := 0, r_0 := 0, r_1 := b - Ax_1$
2: **for** $k = 0, 1, 2, \ldots$ until convergence **do**
3: $v_{sk+1} := r_{sk+1}$
4: Compute MPK $\underline{V}_k = [v_{sk+1}, \ldots, v_{sk+s+1}]$
5: **for** $j = 1$ to s **do** P2P Comm.
6: Compute d_{sk+j} $(Ar_{sk+j} = [R_{k-1}, \underline{V}_k]d_{sk+j})$
7: $w_{sk+j} := [R_{k-1}, \underline{V}_k]d_{sk+j}$
8: $\mu_{sk+j} := \langle r_{sk+j}, r_{sk+j} \rangle$; test for convergence
9: $\nu_{sk+j} := \langle w_{sk+j}, r_{sk+j} \rangle$ Collective Comm.
10: $\gamma_{sk+j} := \mu_{sk+j} / \nu_{sk+j}$
11: **if** $sk + j = 1$ **then**
12: $\rho_{sk+j} := 1$
13: **else**
14: $\rho_{sk+j} := (1 - \frac{\gamma_{sk+j}}{\gamma_{sk+j-1}} \cdot \frac{\mu_{sk+j}}{\mu_{sk+j-1}} \cdot \frac{1}{\rho_{sk+j-1}})^{-1}$
15: **end if**
16: $x_{sk+j+1} := \rho_{sk+j}(x_{sk+j} + \gamma_{sk+j} r_{sk+j}) + (1 - \rho_{sk+j}) x_{sk+j-1}$
17: $r_{sk+j+1} := \rho_{sk+j}(r_{sk+j} + \gamma_{sk+j} w_{sk+j}) + (1 - \rho_{sk+j}) r_{sk+j-1}$
18: **end for**
19: **end for**

Communication-Avoiding CG method

Review of CA-CG method based on [Hoemmen,PhD2010]

- CG

dependency between r_j and p_j

- CG with 3-term recurrence

use only r_j (decouple r_j-p_j dependency)

- CG with outer/inner loops
- CG with Matrix Power Kernel

compute Ar_{sk+j} based on recurrence formula using basis vectors given by MPK

→CA for halo comm.

- CA-CG

compute inner product using Gram matrix

→CA for reduction comm.

CA-CG is equivalent to CG in exact arithmetic

Algorithm 5 CA-CG

Input: $Ax = b$, Initial guess x_1

Output: Approximate solution x_{sk+j}

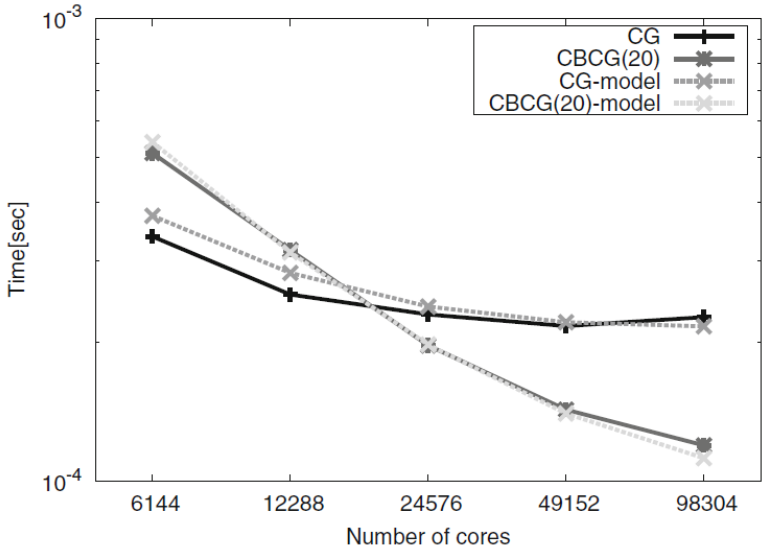
1: $x_0 := 0, r_0 := 0, r_1 := b - Ax_1$
2: **for** $k = 0, 1, 2, \ldots$ until convergence **do**
3: $v_{sk+1} := r_{sk+1}$ — P2P Comm.
4: Compute MPK $\underline{V}_k = [v_{sk+1}, \ldots, v_{sk+s+1}]$
5: $G_{k,k-1} := R^*_{k-1}\underline{V}_k, G_{kk} := \underline{V}^*_k\underline{V}_k$ — Collective Comm.
6: $G_k = \begin{pmatrix} D_{k-1} & G_{k,k-1} \\ G^*_{k,k-1} & G_{kk} \end{pmatrix}$ — Gram Matrix
7: **for** $j = 1$ to s **do**
8: Compute d_{sk+j} $(Ar_{sk+j} = [R_{k-1}, \underline{V}_k]d_{sk+j})$
9: Compute g_{sk+j} $(r_{sk+j} = [R_{k-1}, \underline{V}_k]g_{sk+j})$
10: $w_{sk+j} := [R_{k-1}, \underline{V}_k]d_{sk+j}$
11: $\mu_{sk+j} := g^*_{sk+j}G_k g_{sk+j}$
12: $\nu_{sk+j} := g^*_{sk+j}G_k d_{sk+j}$
13: $\gamma_{sk+j} := \mu_{sk+j}/\nu_{sk+j}$
14: **if** $sk + j = 1$ **then**
15: $\rho_{sk+j} := 1$
16: **else**
17: $\rho_{sk+j} := (1 - \frac{\gamma_{sk+j}}{\gamma_{sk+j-1}} \cdot \frac{\mu_{sk+j}}{\mu_{sk+j-1}} \cdot \frac{1}{\rho_{sk+j-1}})^{-1}$
18: **end if**
19: $x_{sk+j+1} := \rho_{sk+j}(x_{sk+j} + \gamma_{sk+j}r_{sk+j}) + (1 - \rho_{sk+j})x_{sk+j-1}$
20: $r_{sk+j+1} := \rho_{sk+j}(r_{sk+j} + \gamma_{sk+j}w_{sk+j}) + (1 - \rho_{sk+j})r_{sk+j-1}$
21: **end for**
22: **end for**

Related works – Application of CA-Krylov methods

- Stability and convergence properties of CA-Krylov methods [Carson,PhD2015]
  - Convergence properties with different basis vectors
  - Improved convergence with residual replacement technique
- CA preconditioners for CA-Krylov methods [Yamazaki,SC14]
  - CA-GMRES implementation on GPUs
  - CA-preconditioning underlap approach
- Chebyshev basis CA-CG on K-computer [Kumagai,PPAM2015]
  - Strong scaling of CBCG up to 100k cores

→Most of former works were successful for
  - large CA steps with s>10
  - No or approximate preconditioning

→We apply LP-CA-CG to ill-conditioned problem which is limited to s=3



Performance of CBCG on K-computer [Kumagai,PPAM2015]

Convergence issue of CA-CG in JUPITER

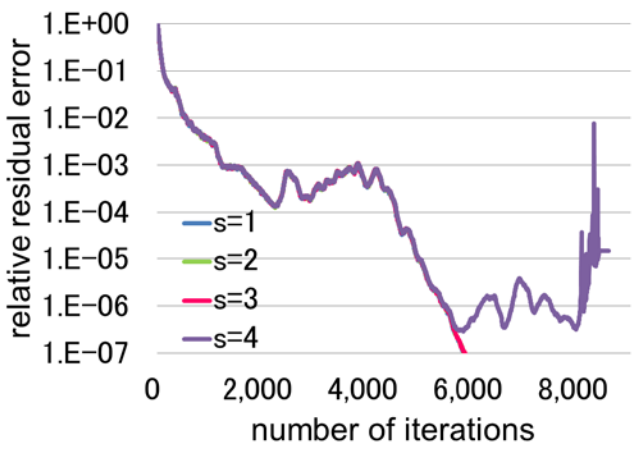
Poor convergence of CA-CG in JUPITER

- CA steps with s>3 do not converge
- Causes of convergence degradation
  - Orthogonality of basis vectors generated during CA steps
  - Round off errors in inner product operations using Gram matrix
- Possible solutions
  - Newton and Chebyshev basis vectors [Hoemmen,PhD2010, Carson,PhD2015]
  - Mixed precision approach

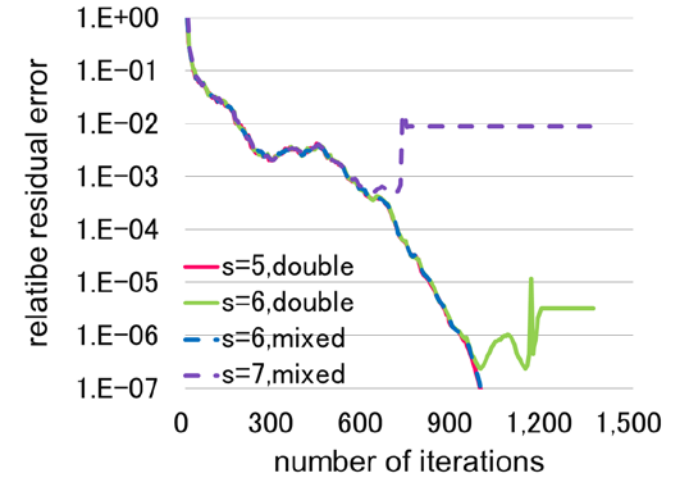
    (quadruple precision only in Gram matrix)

  →Preliminary tests did not show large performance gain

→We pursue performance improvement at s=3



Convergence of CA-CG
(JUPITER:800x500x3540)



Convergence of CA-CG with mixed precision
(JUPITER:104x104x265)

Optimization of CA-CG on K-computer

- Serial optimization
  - Avoid indirect data access by changing data format from CSR to CDR
  - Minimize memory access by data blocking and loop splitting

  →Maximize impact of CA by minimizing cost of calculation

- Comparisons of two Block-Jacobi based preconditioners
  - Original Block-Jacobi (bj) preconditioning
  - CA-preconditioning with underlap (u) approach [Yamazaki,SC14]

  →Explore preconditioners suitable for CA-CG on K-computer

Optimize data format for JUPITER

- Compressed Sparse Row (CSR) format
  - Widely used in many matrix libraries such as PETSc
  - Indirect memory access is overhead for structured grid data
- Compressed Diagonal Storage (CDS) format
  - Used in our in-house solvers
  - Direct memory access for structured grid data

→ 3.3x performance gain compared to CSR format

Sample matrix

$$\begin{pmatrix} 11 & 12 & 0 & 0 \\ 21 & 22 & 23 & 0 \\ 0 & 32 & 33 & 34 \\ 0 & 0 & 43 & 44 \end{pmatrix}$$

CSR format

Pointer table

1	3	6	9

Column index

1	2	1	2	3	2	3	4	3	4

Coefficients

11	12	21	22	23	32	33	34	43	44

SpMV sample code

```
for(i=0; i<n; i++){
    for(i2=ia[i]; i2<ia[i+1]; i2++){
        q[i] =q[i] + a[i2] * p[ja[i2]];}}
```

indirect access

CDS format

Offset from diagonal

−1	0	1

Coefficients

0	21	32	43	11	22	33	44	12	23	34	0

SpMV sample code

```
for(j=0;j<ndia;j++){offset=index[j];
    for(i=0;i<n;i++){
        q[i]=q[i]+a[i+j*n]*p[i+offset];}}
```

Analysis of arithmetic intensity of CA-CG kernels

- Outer loop: SpMV and Block-Jacobi preconditioning (62%)

	SpMV	BJ-precond.
Arithmetic intensity (f/b)	0.163	0.116
Roofline (Gflops)	7.15	5.22
Sustained (Gflops)	6.37	4.99

2nd order centered finite difference with 7-stencils
→ low arithmetic intensity

- Outer loop: Gram matrix computation (10%)

	s=1	s=2	s=3
Arithmetic intensity (f/b)	0.300	0.469	0.636
Roofline (Gflops)	12.60	18.66	24.08
Sustained (Gflops)	14.35	23.60	27.16

$$\frac{f}{b} = \frac{(s+1)\times(2s+1)\times 2}{(3s+3)\times 8} \propto O(s)$$

- Inner loop: Inner product and 3-term recurrence (28%)

	s=1	s=2	s=3
Arithmetic intensity (f/b)	0.105	0.118	0.130
Roofline (Gflops)	4.39	5.13	5.70
Sustained (Gflops)	3.33	3.67	4.32

optimized

s=1	s=2	s=3
0.188	0.354	0.521
8.18	14.62	20.40
7.30	12.85	19.14

$$\frac{f}{b} = \frac{((4s+1)\times 2+15)\times s+2}{((22+(2s+3)\times 2)\times 8)\times s} \propto const.$$

$$\frac{f}{b} = \frac{((4s+1)\times 2+15)\times s+2}{((2s+6)\times 2+2s)\times 8} \propto O(s)$$

Improve arithmetic intensity of inner loop

- Original

```
for j=1, s, j++
  Compute dj, μj, νj, ρj, γj
  for i=1, n, i++
    for k=1, 2s+1, k++
      uj[i] = {Q, V}[i, k]dj[k]
      yj[i] = {Z, W}[i, k]dj[k]
    endfor
    xj+1[i]=ρj(xj[i]+ γjqj[i])+(1−ρj) xj−1[i]
    qj+1[i]=ρj(qj[i]−γjuj[i])+(1−ρj) qj−1[i]
    zj+1[i]=ρj(zj[i]−γjyj[i])+(1−ρj) zj−1[i]
  endfor
endfor
```

{Q,V} and {Z,W} are loaded s times

Reuse {Q,V} and {Z,W} for s times via loop splitting and data blocking

→Reduce memory access from s^2 to s

- Optimized

```
for j=1 ,s ,j++
  Compute dj, μj, νj, ρj, γj
endfor
for ii=1 ,n ,ii=ii+nblock
  for j=1 ,s ,j++
    for i=ii ,ii+nblock-1 ,i++
      for k=1, 2s+1, k++
        uj[i] = {Q, V}[i, k]dj[k]
        yj[i] = {Z, W}[i, k]dj[k]
      endfor
    endfor
  endfor
  for j=1 ,s ,j++
    for i=ii ,ii+nblock-1 ,i++
      xj+1[i]=ρj(xj[i]+ γjqj[i])+(1−ρj) xj−1[i]
      qj+1[i]=ρj(qj[i]−γjuj[i])+(1−ρj) qj−1[i]
      zj+1[i]=ρj(zj[i]−γjyj[i])+(1−ρj) zj−1[i]
    endfor
  endfor
endfor
```

Block-Jacobi based preconditioners for CA-CG

- Original Block-Jacobi preconditioning with SpMV
  - P2P comm. of halo data at every step
  - No additional comm. for BJ preconditioning

  →Hybrid CA-CG method
  (CA inner product + no-CA SpMV)

- Block-Jacobi preconditioning with CA-SpMV
  - Halo data for s-steps is transferred in advance
  - Additional computation for extended halo data

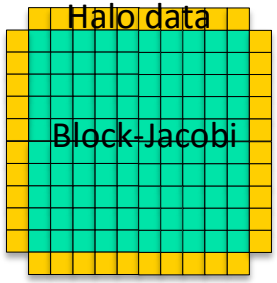
  →Additional P2P comm. for BJ preconditioning

- Underlap preconditioning with CA-SpMV
  - Point-Jacobi preconditioning for surface part

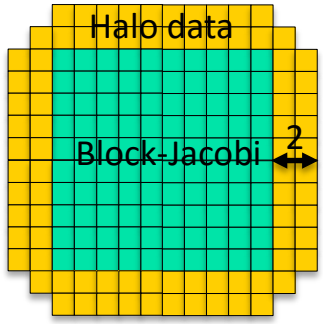
  →CA step and parallelization affect convergence

→Explore optimum preconditioner for K-computer

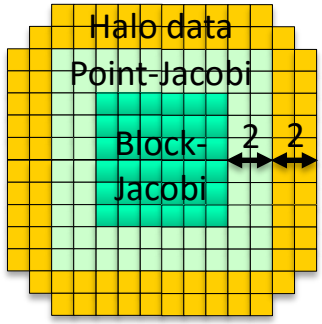
Block-Jacobi with SpMV



Block-Jacobi with CA-SpMV(s=2)



Underlap with CA-SpMV(s=2)

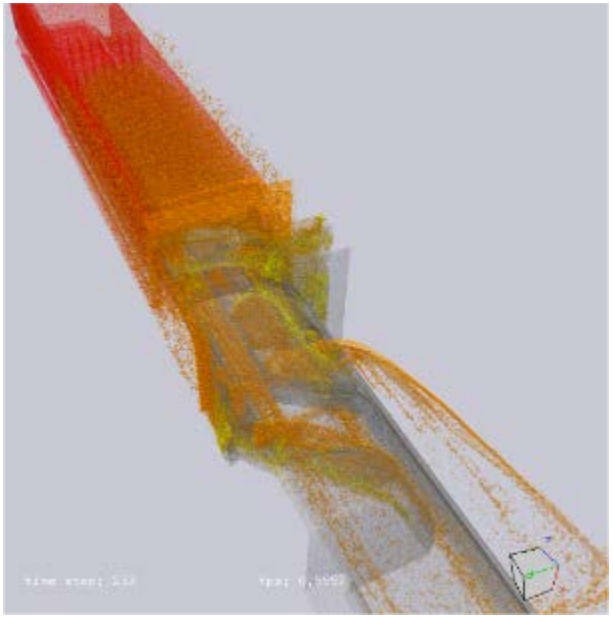


Converge property

- Matrix data from Poisson solver in JUPITER
  - Melt relocation of a fuel assembly
  - Problem size: n = 800 x 500 x 3,540 =1.4 x 10^9

- Comparisons of CG solvers
  - Original CG
  - Hybrid CA-CG with Block-Jacobi (bj)
  - CA-CG with underlap (u)

Visualization of JUPITER test problem



CA step scan of number of iteration at 125 nodes

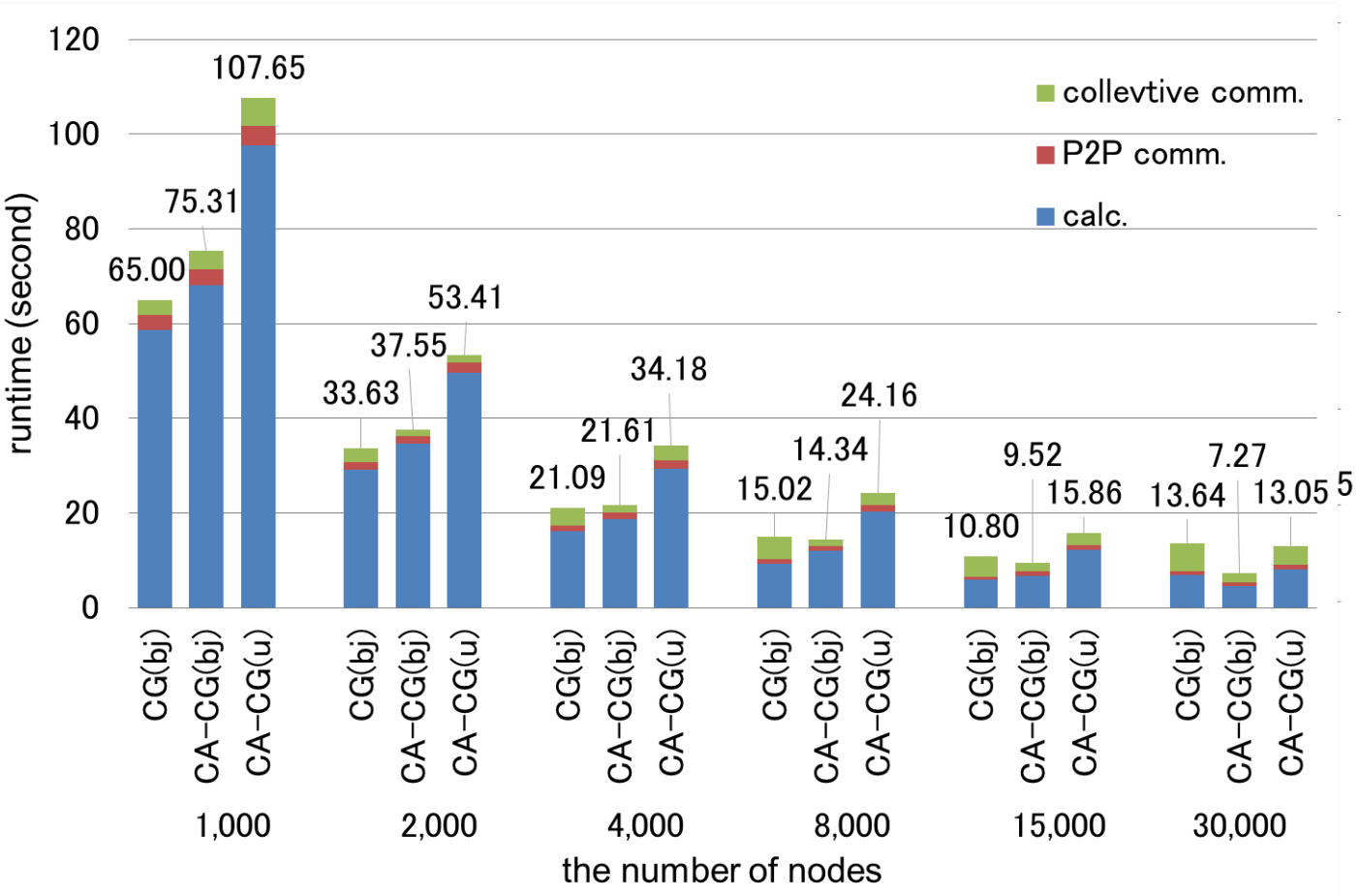
CA steps	s=1	s=2	s=3	s=4
CG(bj)	6206	-	-	-
CA-CG(bj)	6208	6224	6222	×
CA-CG(u)	6214	7188	7668	8072

Node number scan of number of iteration at s=3

Nodes	1000	2000	4000	8000
CG(bj)	6333	6428	6313	6633
CA-CG(bj)	6354	6444	6516	6636
CA-CG(u)	8274	8556	9582	10206

- Impact of CA step and parallelization on Block-Jacobi is weak
- Convergence of underlap degrades with CA step and number of nodes because of expansion of surface part with Point-Jacobi approximation

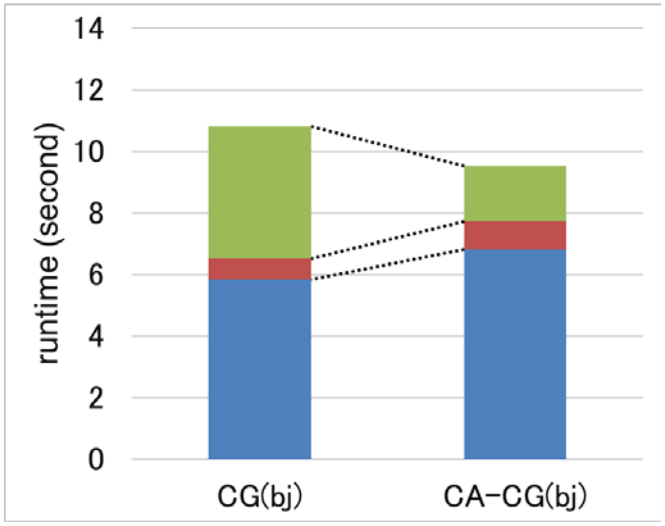
Strong scaling of CA-CG on K-computer



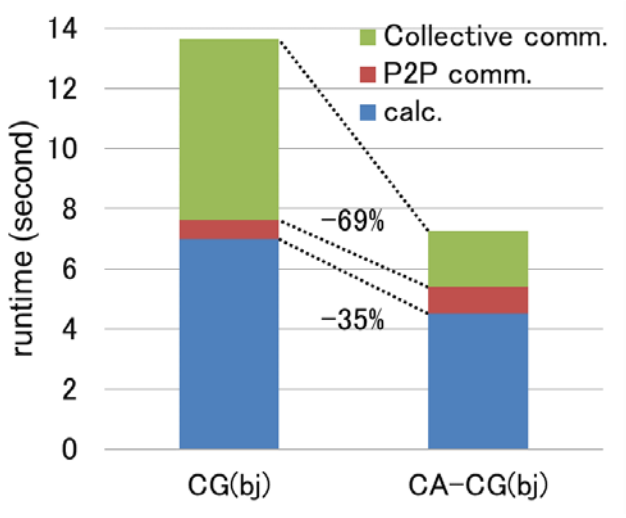
- CA-CG solvers show good strong scaling up to 30,000 nodes (240k cores)
- Hybrid CA-CG(bj) is fastest, and total cost is reduced by 47% from CG(bj)
- In CA-CG(u), advantage of CA is almost cancelled by worse convergence

Detailed cost distribution in CG and hybrid CA-CG

Runtime(s) at 15,000 nodes



Runtime at 30,000 nodes



- Collective communication is reduced by 1/s=1/3
- P2P communication is almost comparable
- Calculation part shows different features between 15k and 30k nodes
  - Up to 15k nodes, calculation cost of CA-CG is slightly higher than CG
    - FP operations/Performance of CA-CG is increased by 2.4x/2x
  - At 30k nodes, calculation cost of CG is lower than CA-CG
    - Inner most loop size is not enough for pipelining

→Compute rich CA-CG can keep performance in such strong scaling limit

Conclusion

- LP-CA-CG method is applied to ill-conditioned pressure Poisson equation in multiphase CFD code JUPITER
  - CA procedures affect convergence and CA steps are limited to s=3
  - Even at s=3, significant performance gain was achieved on K-computer
    - Optimum data format for structured grid data
    - Improved arithmetic intensity with loop splitting implementation
    - Hybrid CA-CG approach based on 3D torus network

    →CA-Krylov solver design strongly depends on computing platforms

Future Work

- Development of CA-Krylov solvers for different computing platforms
- Improvement of CA-Krylov methods

  (e.g. preconditioning, basis vectors, residual replacement, mixed-precision)