Recent Challenges of Auto-tuning: Accuracy Optimization and Explainable AI

Takahiro Katagiri

(Information Technology Center, Nagoya University)

International Workshop on the Integration of (Simulation + Data + Learning) Towards Society 5.0 by h3-Open-BDEC, Nov. 30 & Dec. 3, 2021
Session 3 (3 Dec 8:40~9:00 JST) Adaptive precision, AT & Verification (II)

Hierarchical, Hybrid, Heterogeneous
h3-Open-BDEC
Big Data & Extreme Computing

名古屋大学
NAGOYA UNIVERSITY

h3-Open-BDEC

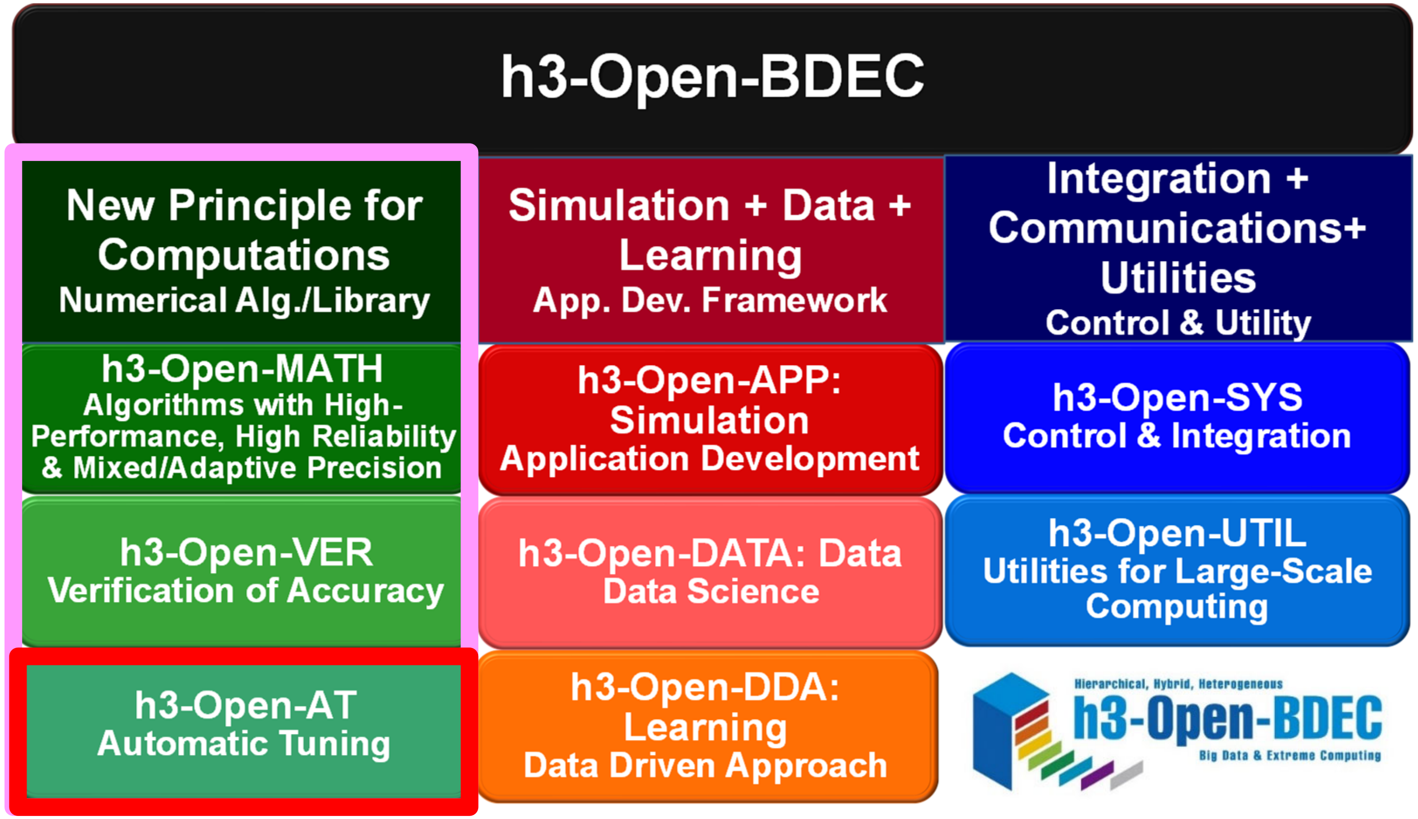
Innovative Software Platform for Integration of (S+D+L) on BDEC

h3-Open-BDEC		
New Principle for Computations Numerical Alg./Library	**Simulation + Data + Learning** App. Dev. Framework	**Integration + Communications+ Utilities** Control & Utility
h3-Open-MATH Algorithms with High-Performance, High Reliability & Mixed/Adaptive Precision	**h3-Open-APP: Simulation** Application Development	**h3-Open-SYS** Control & Integration
h3-Open-VER Verification of Accuracy	**h3-Open-DATA: Data** Data Science	**h3-Open-UTIL** Utilities for Large-Scale Computing
h3-Open-AT Automatic Tuning	**h3-Open-DDA: Learning** Data Driven Approach	Hierarchical, Hybrid, Heterogeneous h3-Open-BDEC Big Data & Extreme Computing

h3-Open-BDEC: Two Significant Innovations

① Methods for Numerical Analysis with High-Performance/High-Reliability/Power-Saving based on the New Principle of Computing by

- ✓ Adaptive Precision
- ✓ Accuracy Verification
- ✓ Automatic Tuning



h3-Open-BDEC

New Principle for Computations
Numerical Alg./Library

h3-Open-MATH
Algorithms with High-Performance, High Reliability & Mixed/Adaptive Precision

h3-Open-VER
Verification of Accuracy

h3-Open-AT
Automatic Tuning

Simulation + Data + Learning
App. Dev. Framework

h3-Open-APP: Simulation
Application Development

h3-Open-DATA: Data
Data Science

h3-Open-DDA: Learning
Data Driven Approach

Integration + Communications+ Utilities
Control & Utility

h3-Open-SYS
Control & Integration

h3-Open-UTIL
Utilities for Large-Scale Computing

Hierarchical, Hybrid, Heterogeneous
h3-Open-BDEC
Big Data & Extreme Computing

Outline

- Main issues in this talk:
  1. How to reduce cost of tuning for mixed-precision computations and/or energy consumption?
     - Answer: Use AT framework!
  2. Is AI result of tuning on numerical libraries reliable?
     - Answer: Use Explainable AI (XAI)! →**Scientific XAI (SXAI)**
- TOPIC I
  - Mixed-Precision and Energy Optimization by ppOpen-AT
- TOPIC II
  - Explainable AI for Auto-tuning on an Accurate Precision Matrix-Matrix Multiplication Library

Outline

- TOPIC I
  - Mixed-Precision and Energy Optimization by ppOpen-AT
- TOPIC II
  - Explainable AI for Auto-tuning on an Accurate Precision Matrix-Matrix Multiplication Library

Collaboration with

- Mr. Shohei Yamanashi, Master Course Student, Graduate School of Informatics, Nagoya University
- Dr. Hisashi Yashiro, Center for Global Environment Research, National Institute for Environmental Studies

A Proposal of Mixed-Precision and/or Energy Optimization for ppOpen-AT

Hierarchical, Hybrid, Heterogeneous
h3-Open-BDEC
Big Data & Extreme Computing

名古屋大学
NAGOYA UNIVERSITY

Background

- More complex computer architectures are being designed toward to era of Post Moore:
  - Multi-cores on CPUs, Deep hierarchies for memory, Low precision computations, Quantum Computing, etc.
- Mixed Precision Computations
  - Low precision computation (single/half) is applied for a part of computations in programs.

⇒Obtaining speedup and low energy.

Aim of this study

- Automation of performance tuning for mixed precision computation by adapting Software Auto-tuning (AT) Technology [I-1].
- The followings are targets in mixed precision computations in this research:
  1. Variables / Arrays
  2. Blocks
  3. Functions / Sub routines
- New directives of AT for the above are proposed for ppOpen-AT.

[I-1] Takahiro Katagiri, Daisuke Takahashi, Japanese Autotuning Research: Autotuning Languages and FFT, Proceedings of the IEEE, Volume: 106, Issue: 11, Nov. 2018, pp. 2056-2067 (2018).

An AT Language: ppOpen-AT[I-2]

- AT language to add AT functions to programs.
- Code generator makes the followings automatically:
  1. Multiple candidates of optimized code.
  2. Search program for AT to find the best candidate.

```
!oat$ install unroll (i) region start
!oat$ varied (i) from 1 to 4
  do i = 1 , n
    do j = 1, n
      do k = 1, n
        A(i, j) = A(i, j) + B(i, k) * C(k, j)
  enddo; enddo; enddo
!oat$ install unroll (i) region end
```

Ex) Loop unrolling from 1st to 4th depths.

Code Generation

```
do i = 1, (n/3)*3, 3
    do  j = 1, n
        do k = 1, n
            A(i,j) = A(i,j) + B(i,k) * C(k,j)
            A(i+1,j) = A(i+1,j)+B(i+1,k)*C(k,j)
            A(i+2,j) = A(i+2,j)+B(i+2,k)*C(k.j)
enddo; enddo; enddo
if (mod(n, 3) /= 0) then
    do i = (n/3)*3+1, n
        do j = 1, n
            do k = 1, n
                A(i, j) = A(i, j) + B(i, k) * C(k, j)
    enddo; enddo; enddo
endif
```

Ex) loop unrolling with 3rd depth.

[I-2] 片桐孝洋, ppOpen-AT: ポストペタスケール時代の数値シミュレーション基盤ソフト,
数理解析研究所講究録 第1791巻, pp. 107-111 (2012).

Process Flow for AT of Mixed-Precision Computations and/or Energy

"Users" are defined by:

- Software Developers;
- End-Users;

The following slides explain each step.

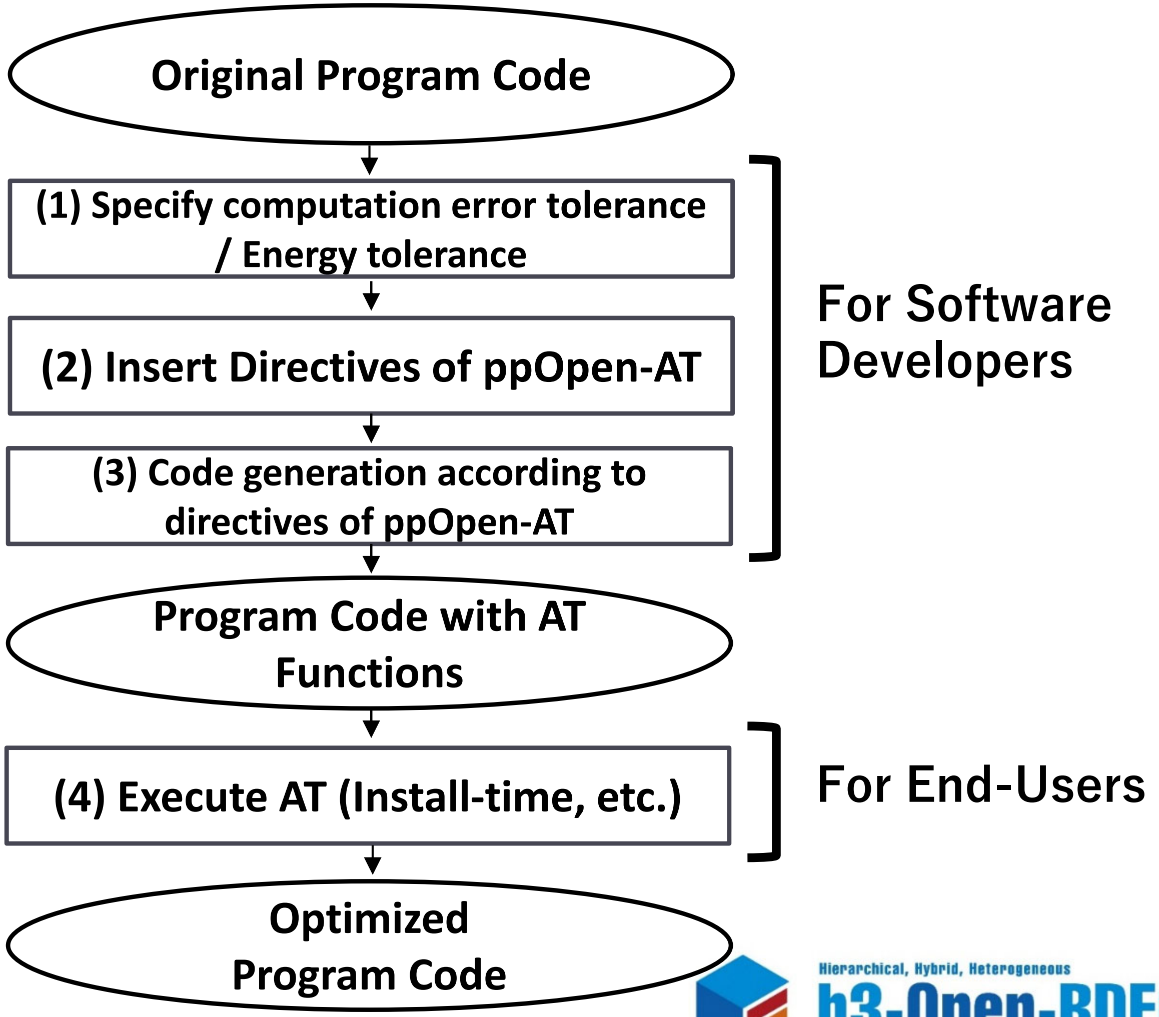
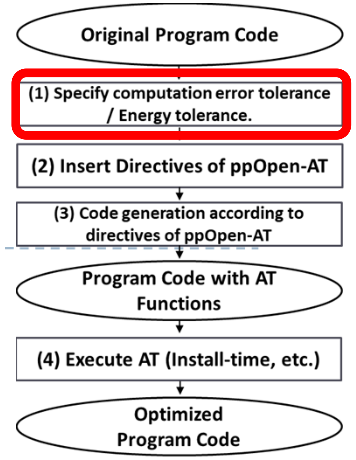

Original Program Code
(1) Specify computation error tolerance / Energy tolerance
(2) Insert Directives of ppOpen-AT
(3) Code generation according to directives of ppOpen-AT
Program Code with AT Functions
(4) Execute AT (Install-time, etc.)
Optimized Program Code
For Software Developers
For End-Users

Fig: Process Flow for the Proposal AT

(1) Specify computation error tolerance / Energy tolerance


Original Program Code

(1) Specify computation error tolerance / Energy tolerance.

(2) Insert Directives of ppOpen-AT

(3) Code generation according to directives of ppOpen-AT

Program Code with AT Functions

(4) Execute AT (Install-time, etc.)

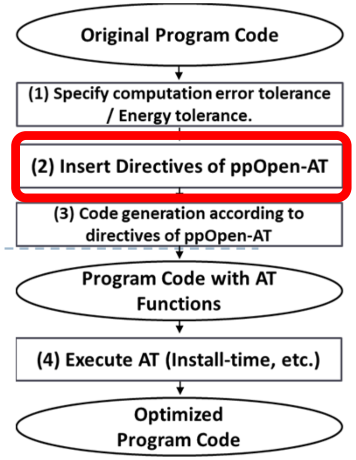
Optimized Program Code

- Software Developers specified the following

by using directives of ppOpen-AT.

- Computation Error Tolerance
  - Specify a tolerance error ratio from original computations (such as double precision) to mixed-precision computations (such as double and single precisions)
  - Ex) **1e-7** in relative error. The AT system tries to optimize program code within 1e-7 in relative error.
- Energy Tolerance
  - Specify a tolerance ratio to energy from original computations (such as doble precision) to mixed-precision computations (such as double and single precisions)
  - Ex) **less than 10%** energy reduction to energy of original code.

(2) Insert Directives of ppOpen-AT


Original Program Code
(1) Specify computation error tolerance / Energy tolerance.
(2) Insert Directives of ppOpen-AT
(3) Code generation according to directives of ppOpen-AT
Program Code with AT Functions
(4) Execute AT (Install-time, etc.)
Optimized Program Code

The following directives is inserted to original program by software developers.

1. Variables / Arrays

```
!oat$ MixedPrecision variables, [clause . . . ]
    {structured-block}
!oat$ end MixedPrecision variables
```

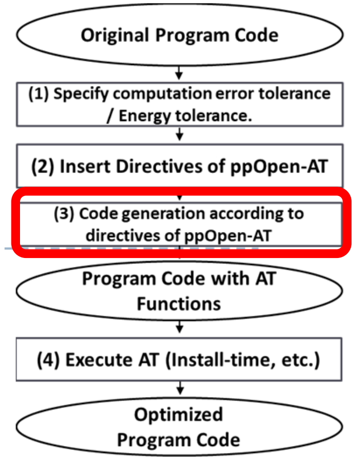
2. Blocks

```
!oat$ MixedPrecision blocks, [clause. . . ]
{
!oat$ MixedPrecision block <num>
       {structured-block}
!oat$ end MixedPrecision  block <num>
 . . .
}
!oat$ end MixedPrecision blocks
```

3. Functions / Sub routines

```
!oat$ MixedPrecision subprogram, [clause . . . ]
       {structured-block}
!oat$ end MixedPrecision subprogram
```

(3) Code generation according to directives of ppOpen-AT


Original Program Code
(1) Specify computation error tolerance / Energy tolerance.
(2) Insert Directives of ppOpen-AT
(3) Code generation according to directives of ppOpen-AT
Program Code with AT Functions
(4) Execute AT (Install-time, etc.)
Optimized Program Code

1. Variables / Arrays

Cantates codes are generated by the directives.

Preparation copies (overhead) are needed.

```
!oat$ MixedPrecision variables, ¥
ChangeVariables(B(:,:),C(:,:)), ¥
ChangePrecision(DP,SP)
do i = 1, n
  do j = 1, n
    do k = 1, n
      B(i, k) = B(i, k) + 2.0_DP
      C(k, j) = C(k, j) + 2.0_DP
      A(i, j) = A(i, j) + B(i, k) * C(k, j)
enddo; enddo; enddo
!oat$ end MixedPrecision variables
```

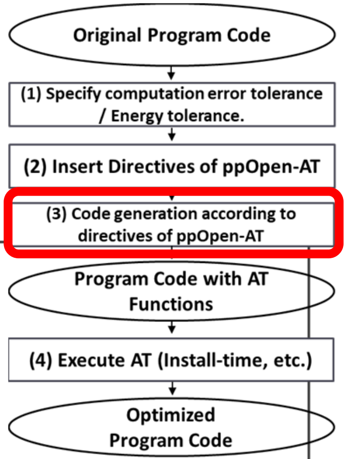
Example of Directives for Variables / Arrays.

Code Generation

```
real(SP) :: B_SP(n,n)
real(SP) :: C_SP(n,n)

B_SP(:,:) = B(:,:)
C_SP(:,:) = C(:,:)

!oat$ MixedPrecision variables, ¥
ChangeVariables(B(:,:),C(:,:)), ¥
ChangePrecision(DP,SP)
do i = 1, n
  do j = 1, n
    do k = 1, n
      B_SP(i, k) = B_SP(i, k) + 2.0_DP
      C_SP(k, j) = C_SP(k, j) + 2.0_DP
      A(i, j) = A(i, j) + B_SP(i, k) * C_SP(k, j)
enddo; enddo; enddo
!oat$ end MixedPrecision variables

B(:,:) = B_SP(:,:)
C(:,:) = C_SP(:,:)
```

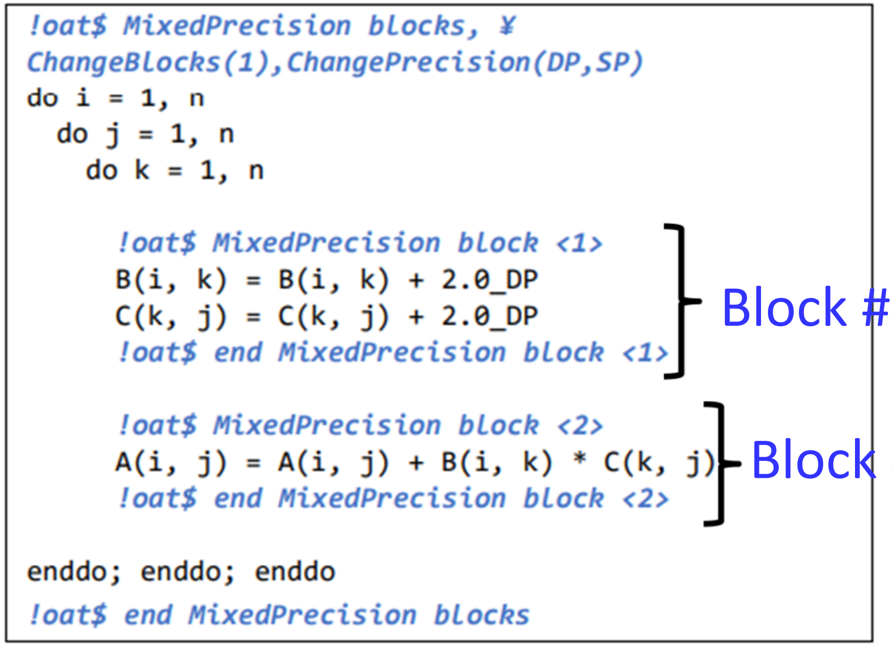
A Candidate Code for Single Precision Computations of Arrays B and C.

(3) Code generation according to directives of ppOpen-AT


Original Program Code
(1) Specify computation error tolerance / Energy tolerance.
(2) Insert Directives of ppOpen-AT
(3) Code generation according to directives of ppOpen-AT
Program Code with AT Functions
(4) Execute AT (Install-time, etc.)
Optimized Program Code

2. Blocks

Cantates codes are generated by the directives.



```
!oat$ MixedPrecision blocks, ¥
ChangeBlocks(1),ChangePrecision(DP,SP)
do i = 1, n
  do j = 1, n
    do k = 1, n

      !oat$ MixedPrecision block <1>
      B(i, k) = B(i, k) + 2.0_DP
      C(k, j) = C(k, j) + 2.0_DP
      !oat$ end MixedPrecision block <1>

      !oat$ MixedPrecision block <2>
      A(i, j) = A(i, j) + B(i, k) * C(k, j)
      !oat$ end MixedPrecision block <2>

enddo; enddo; enddo
!oat$ end MixedPrecision blocks
```

Block #1

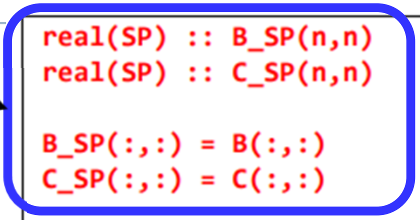
Block #2

Example of Directives for Blocks.

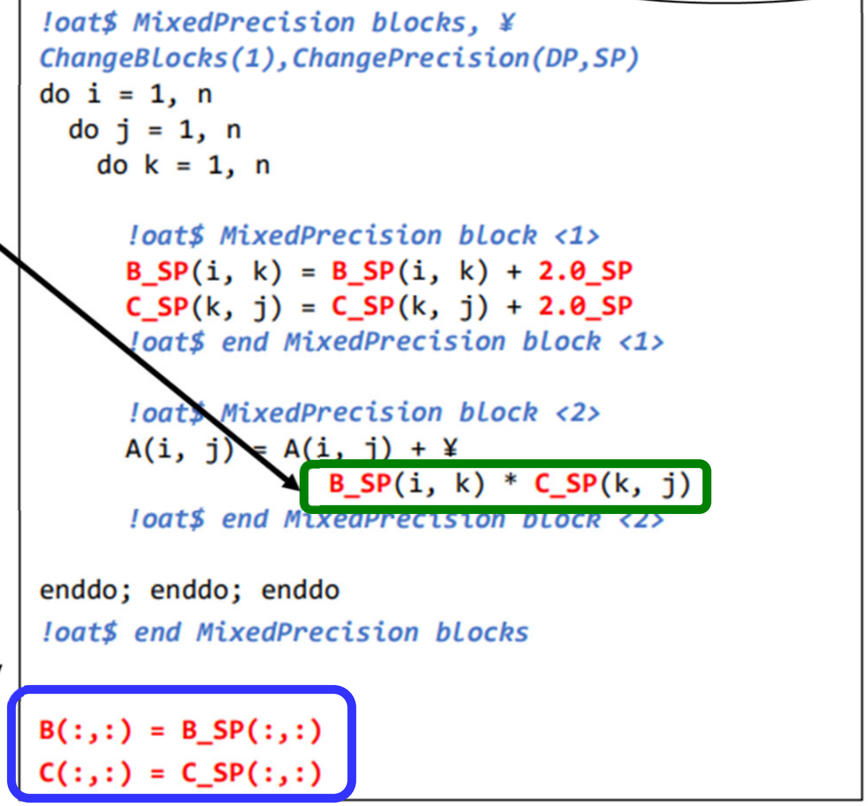
Preparation copies (overhead) are needed.

Propagation Sigle-lization

Code Generation



```
real(SP) :: B_SP(n,n)
real(SP) :: C_SP(n,n)

B_SP(:,:) = B(:,:)
C_SP(:,:) = C(:,:)
```



```
!oat$ MixedPrecision blocks, ¥
ChangeBlocks(1),ChangePrecision(DP,SP)
do i = 1, n
  do j = 1, n
    do k = 1, n

      !oat$ MixedPrecision block <1>
      B_SP(i, k) = B_SP(i, k) + 2.0_SP
      C_SP(k, j) = C_SP(k, j) + 2.0_SP
      !oat$ end MixedPrecision block <1>

      !oat$ MixedPrecision block <2>
      A(i, j) = A(i, j) + ¥
        B_SP(i, k) * C_SP(k, j)
      !oat$ end MixedPrecision block <2>

enddo; enddo; enddo
!oat$ end MixedPrecision blocks

B(:,:) = B_SP(:,:)
C(:,:) = C_SP(:,:)
```

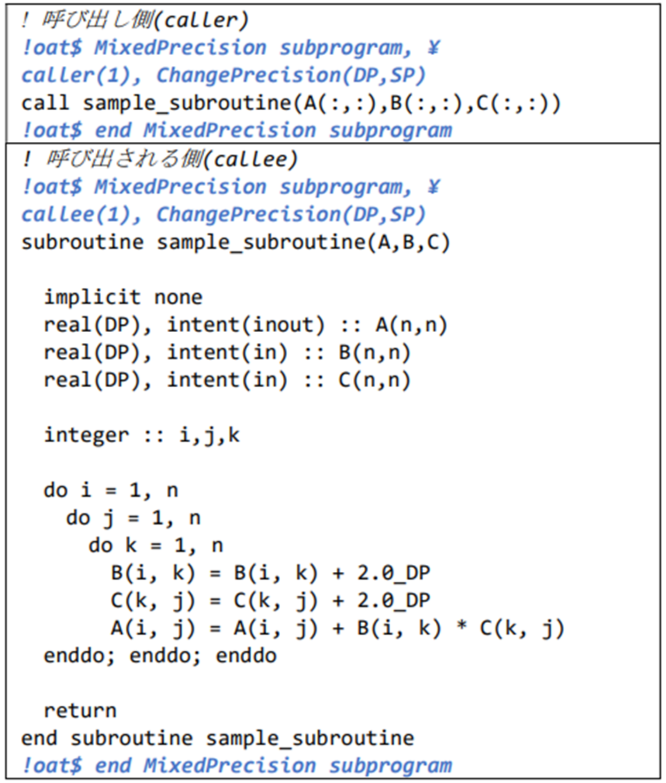
A Candidate Code
for Single Precision Computations
of Block#1

(3) Code generation according to directives of ppOpen-AT

3. Functions/Subroutines

Cantates codes are

generated by the directives.



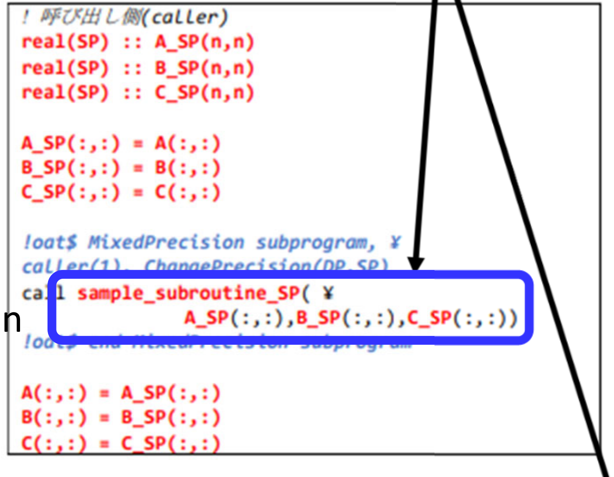
```
! 呼び出し側(caller)
!oat$ MixedPrecision subprogram, ¥
caller(1), ChangePrecision(DP,SP)
call sample_subroutine(A(:,:),B(:,:),C(:,:))
!oat$ end MixedPrecision subprogram
```

```
! 呼び出される側(callee)
!oat$ MixedPrecision subprogram, ¥
callee(1), ChangePrecision(DP,SP)
subroutine sample_subroutine(A,B,C)

  implicit none
  real(DP), intent(inout) :: A(n,n)
  real(DP), intent(in) :: B(n,n)
  real(DP), intent(in) :: C(n,n)

  integer :: i,j,k

  do i = 1, n
    do j = 1, n
      do k = 1, n
        B(i, k) = B(i, k) + 2.0_DP
        C(k, j) = C(k, j) + 2.0_DP
        A(i, j) = A(i, j) + B(i, k) * C(k, j)
  enddo; enddo; enddo

  return
end subroutine sample_subroutine
!oat$ end MixedPrecision subprogram
```

Example of Directives for Subroutines.

Code Generation

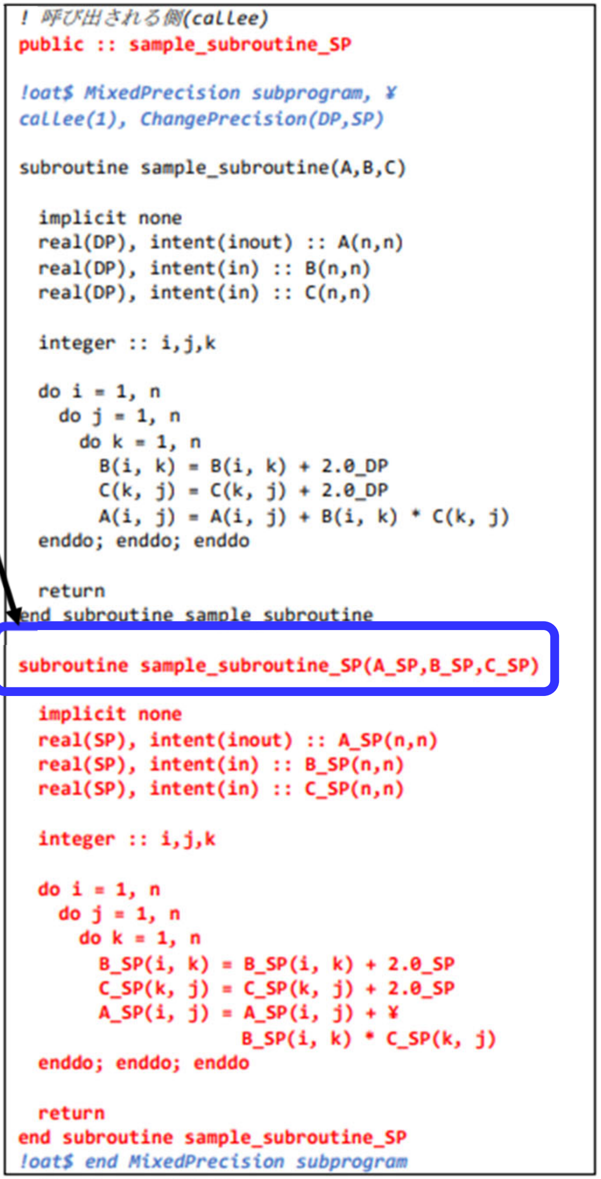
Subroutine name is re-named with single precision.



```
! 呼び出し側(caller)
real(SP) :: A_SP(n,n)
real(SP) :: B_SP(n,n)
real(SP) :: C_SP(n,n)

A_SP(:,:) = A(:,:)
B_SP(:,:) = B(:,:)
C_SP(:,:) = C(:,:)

!oat$ MixedPrecision subprogram, ¥
caller(1), ChangePrecision(DP,SP)
call sample_subroutine_SP( ¥
             A_SP(:,:),B_SP(:,:),C_SP(:,:))
!oat$ end MixedPrecision subprogram

A(:,:) = A_SP(:,:)
B(:,:) = B_SP(:,:)
C(:,:) = C_SP(:,:)
```

Candidate Code (Caller)

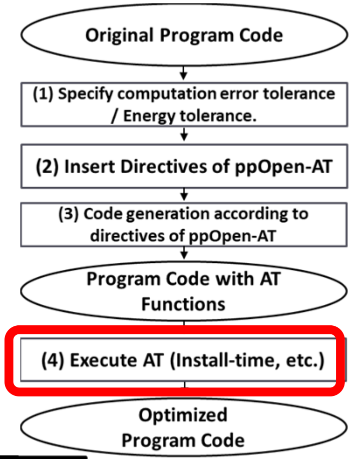
Candidate Code (Callee)



```
! 呼び出される側(callee)
public :: sample_subroutine_SP

!oat$ MixedPrecision subprogram, ¥
callee(1), ChangePrecision(DP,SP)

subroutine sample_subroutine(A,B,C)

  implicit none
  real(DP), intent(inout) :: A(n,n)
  real(DP), intent(in) :: B(n,n)
  real(DP), intent(in) :: C(n,n)

  integer :: i,j,k

  do i = 1, n
    do j = 1, n
      do k = 1, n
        B(i, k) = B(i, k) + 2.0_DP
        C(k, j) = C(k, j) + 2.0_DP
        A(i, j) = A(i, j) + B(i, k) * C(k, j)
  enddo; enddo; enddo

  return
end subroutine sample_subroutine

subroutine sample_subroutine_SP(A_SP,B_SP,C_SP)

  implicit none
  real(SP), intent(inout) :: A_SP(n,n)
  real(SP), intent(in) :: B_SP(n,n)
  real(SP), intent(in) :: C_SP(n,n)

  integer :: i,j,k

  do i = 1, n
    do j = 1, n
      do k = 1, n
        B_SP(i, k) = B_SP(i, k) + 2.0_SP
        C_SP(k, j) = C_SP(k, j) + 2.0_SP
        A_SP(i, j) = A_SP(i, j) + ¥
                     B_SP(i, k) * C_SP(k, j)
  enddo; enddo; enddo

  return
end subroutine sample_subroutine_SP
!oat$ end MixedPrecision subprogram
```

Code Generation

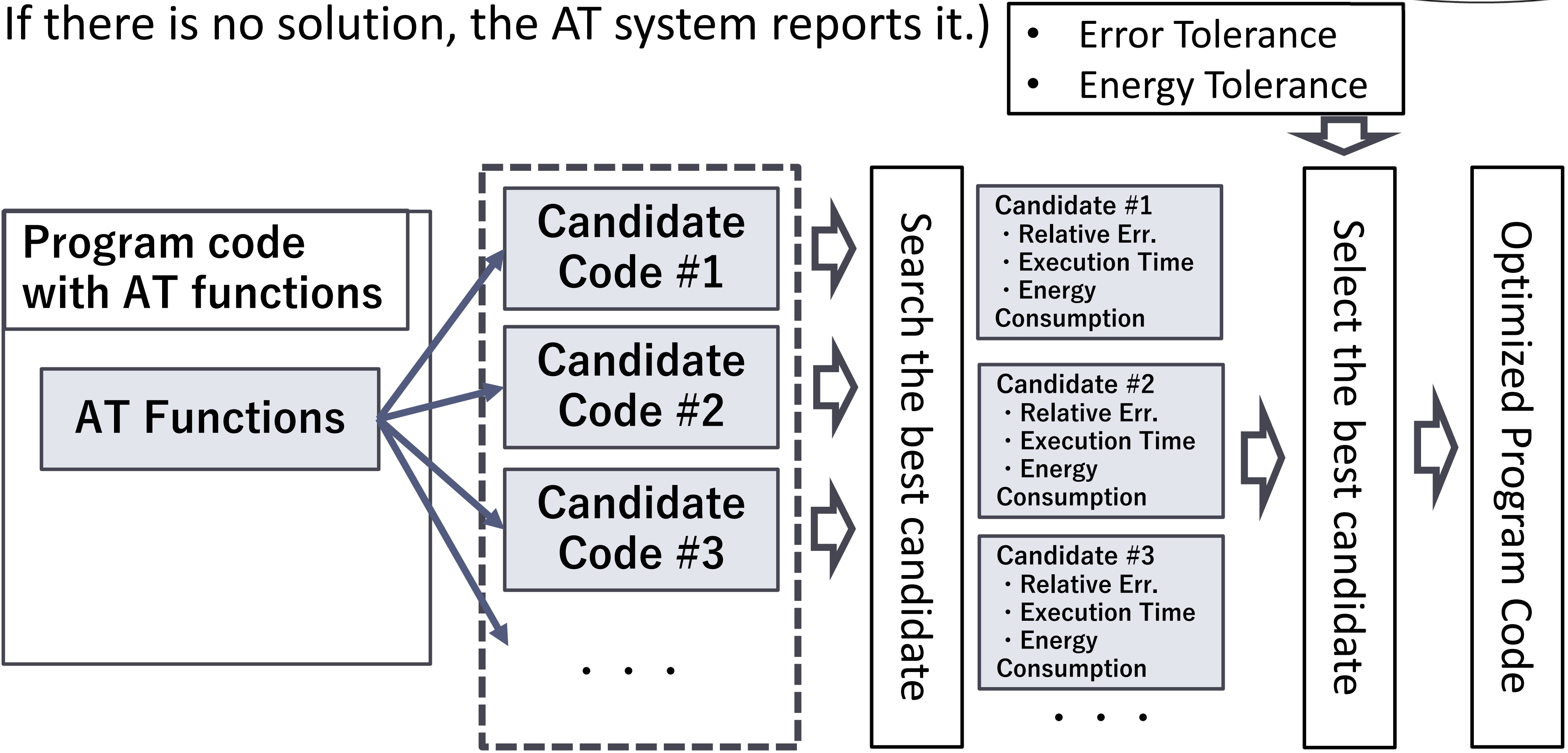
(4) Execute AT (Install-time, etc.)


Original Program Code
(1) Specify computation error tolerance / Energy tolerance.
(2) Insert Directives of ppOpen-AT
(3) Code generation according to directives of ppOpen-AT
Program Code with AT Functions
(4) Execute AT (Install-time, etc.)
Optimized Program Code

End-Users perform AT, like Install-time, etc.

In the AT, system tries to find the best candidate based on error tolerance and/or energy tolerance in the process (1).

(If there is no solution, the AT system reports it.)

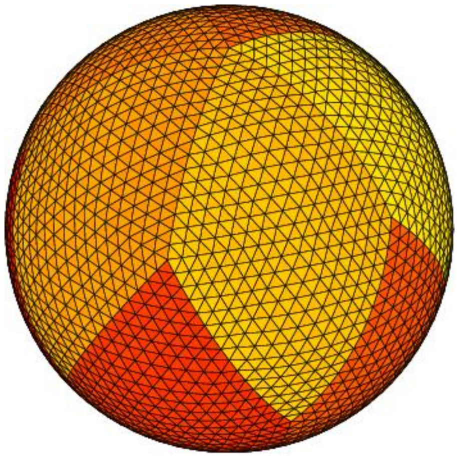


Target Application

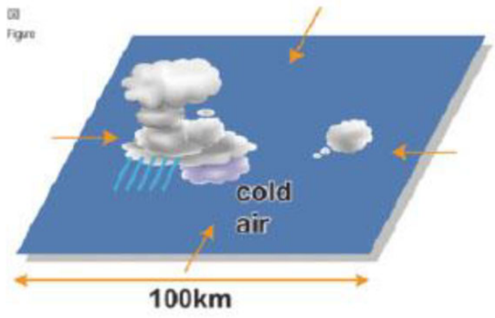
■ Global cloud resolution model NICAM[I-3]

Non-hydrostatic ICosahedral Atmospheric Model.

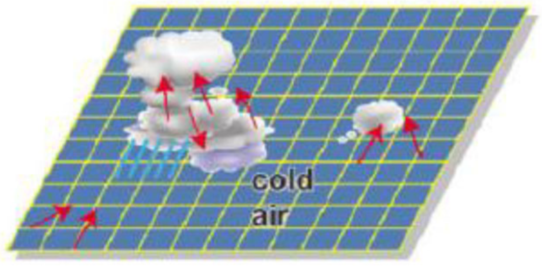
Computational grids for earth atmosphere is introduced, then elements of weather factors are calculated in each grid.



Source: What is NICAM? https://cesd.aori.u-tokyo.ac.jp/satoh/nicam_common.html



Conventional Model



NICAM Model

Source: 富田浩文,京を使った高解像度全球大気シミュレーションの成果とこれからの展望

[I-3] Satoh, M., Tomita, H., Yashiro, H. et al. The Non-hydrostatic Icosahedral Atmospheric Model: description and development. Prog. in Earth and Planet. Sci. 1, 18 (2014).

Target Program

- nicam_dckernel_2016 [I-3]

  : A package of benchmark for NICAM

  - A subroutine **mod_mp_nsw6.f90**, in physicskernel_microphysics on nicam_dckernel_2016.
    - **Physical computation of tiny cloud.**
- Characteristics of the target program
  - **Very long loop body** in the **three-nested loop**.
  - Loop count is fixed.
  - Output values are calculated inside the loop
    - This indicates: sensitive for low precision computations for the output values.

[I-3] https://github.com/hisashiyashiro/nicam_dckernel_2016

Condition of The Experiment

- 1. Valuables/Arrays, and 2. Blocks, are evaluated.

1. Valuables/Arrays
   - 183 valuables/arrays in the target programs are grouping into 13 groups.
   - Low precision-nize (Double to Single) are done in each group .
2. Blocks
   - Target blocks in the target programs are split to 36 blocks.
   - Low precision-nize (Double to Single) are done in each block .

Details of the Experiment

- Execution Time
  - The target three-nested loops

- Maximum Relative Error

$$max\left\{\left|\frac{Output\ of\ DP - Output\ of\ changing\ computational\ accuracy}{Output\ of\ DP}\right|\right\}$$

- Energy Consumption
  - Energy of the target three-nested loop is measured.
  - PowerAPI [5] by Fujitsu Ltd. is used.

[5] 富士通，“FUJITSU Software Technical Computing Suite V4.0L20　ジョブ運用ソフトウェア　APIユーザーズガイド　Power API編”，J2UL-2462-02Z0(01)，2020年6月

Computer Environment

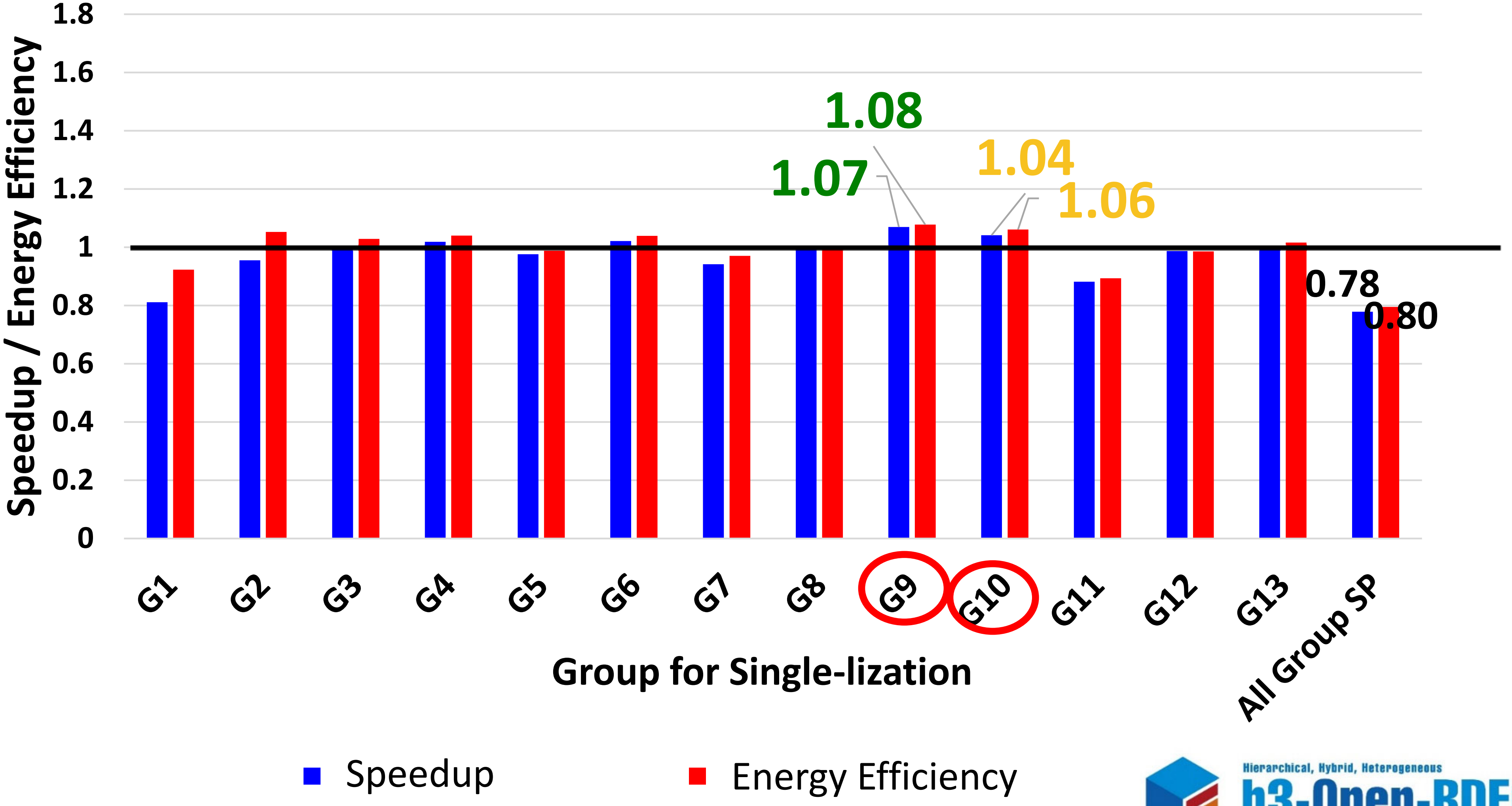
The Supercomputer “Flow” Type I Subsystem, Information Technology Center, Nagoya University

- “Fugaku” Type supercomputer

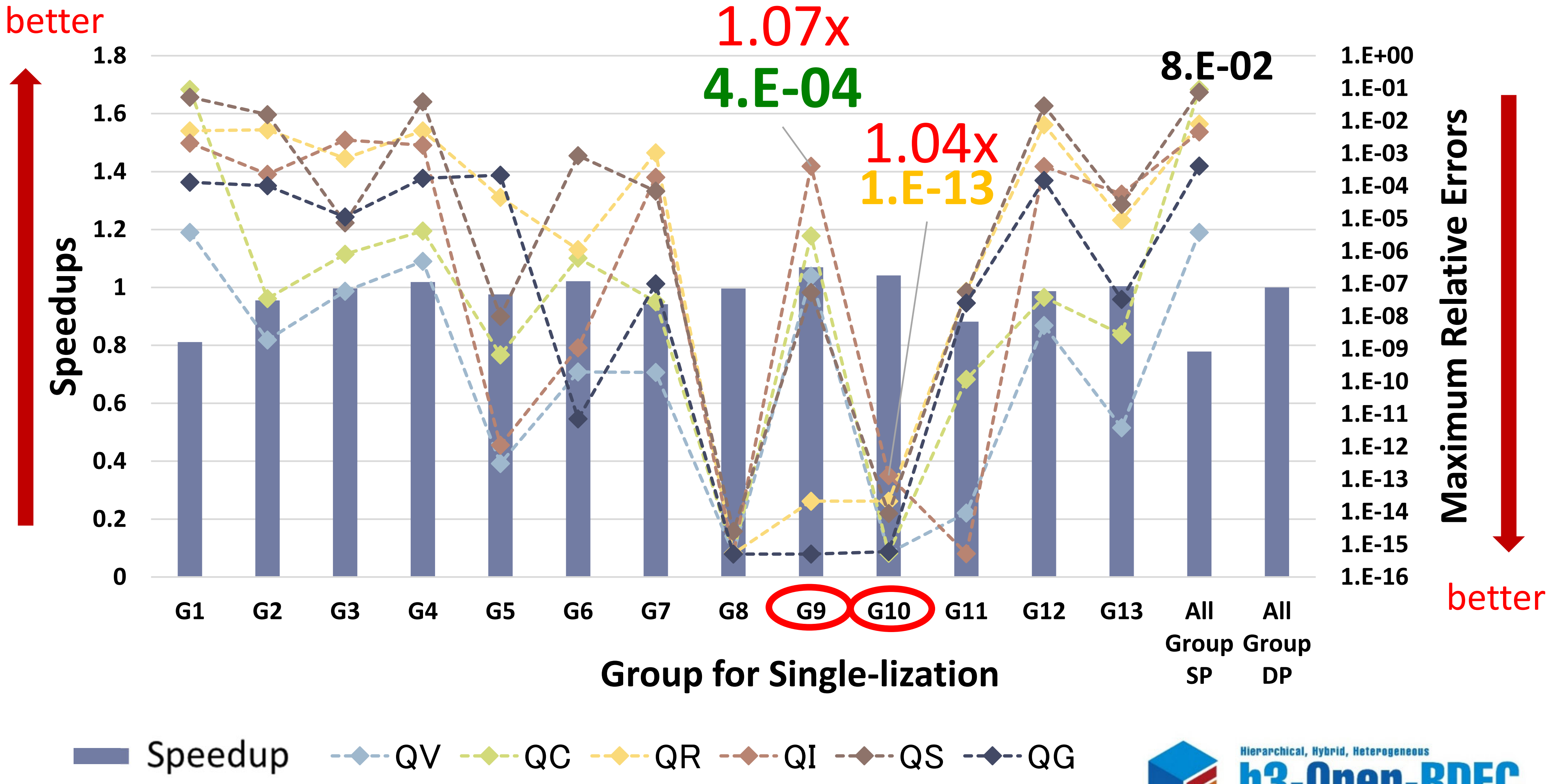
FUJITSU Supercomputer PRIMEHPC FX1000	
Processor	A64FX (Armv8.2-A + SVE)
Number of Processor per Node	1
Number of Cores per Node	48 Cores + 2 Assistant Cores
Frequency	2.2GHz
Theoretical Performance per Node	Double precision: 3.3792 TFLOPS Single precision: 6.7584 TFLOPS Half precision: 13.5168 TFLOPS

Software	
C	frtpx: Fujitsu Fortran Compiler 4.5.0 tcsds-1.2.31
Compiler options	Kfast,ocl,preex,noalias=s,mfunc=2 -Nlst=i -Nlst=t -X03 -Ncompdisp -Koptmsg=2 –Cpp -Kdynamic_iteration -Ksimd,openmp -Kauto,threadsafe -x- -Kprefetch_sequential=soft –Nclang -L /opt/FJSVtcs/pwrm/aarch64/lib64 -lpwr

Result of 1. Variables / Arrays: Speedup and Energy Reduction

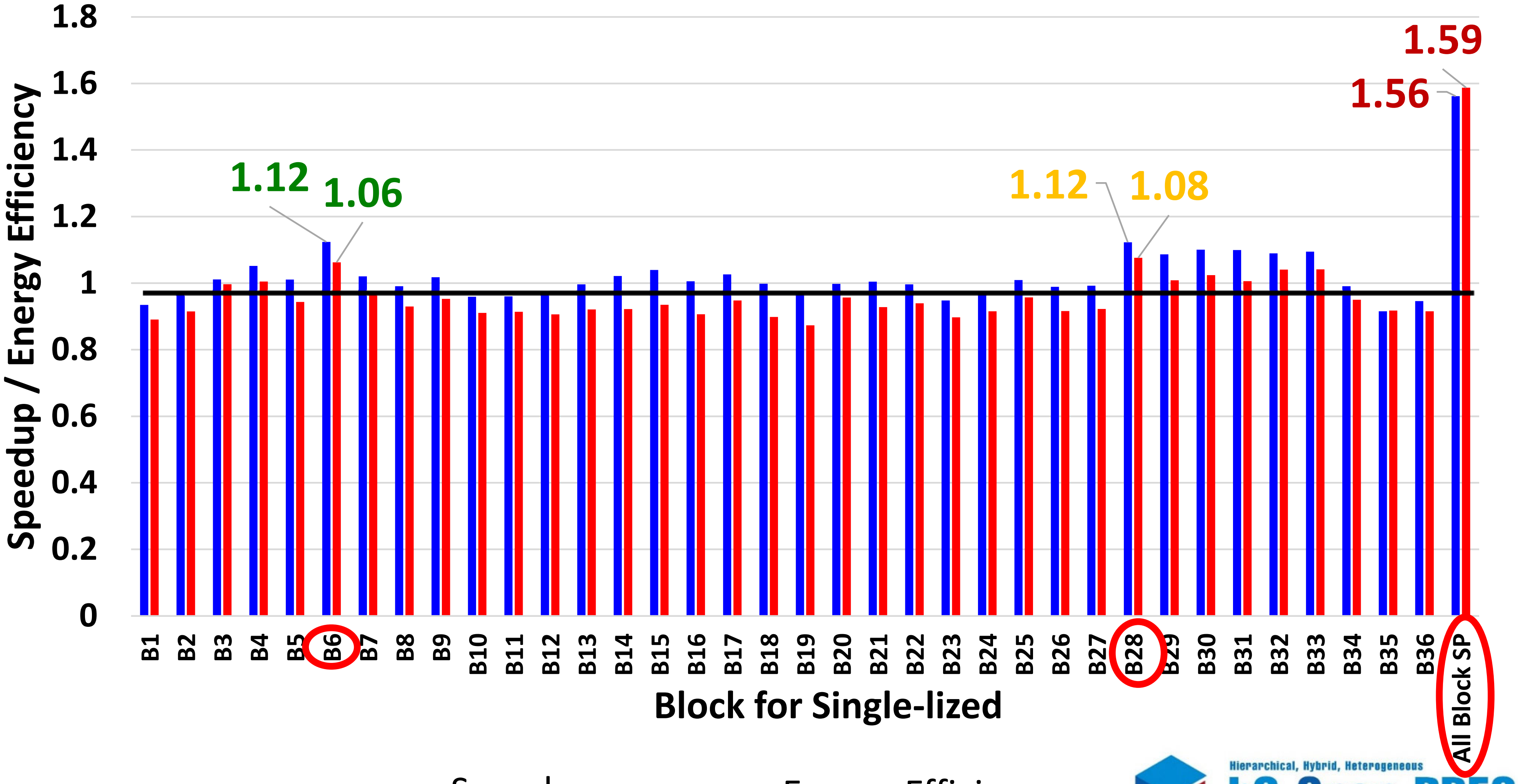


Result of 1. Variables / Arrays: Speedup and Maximum Relative Errors



Checksum Arrays (6 Kinds)

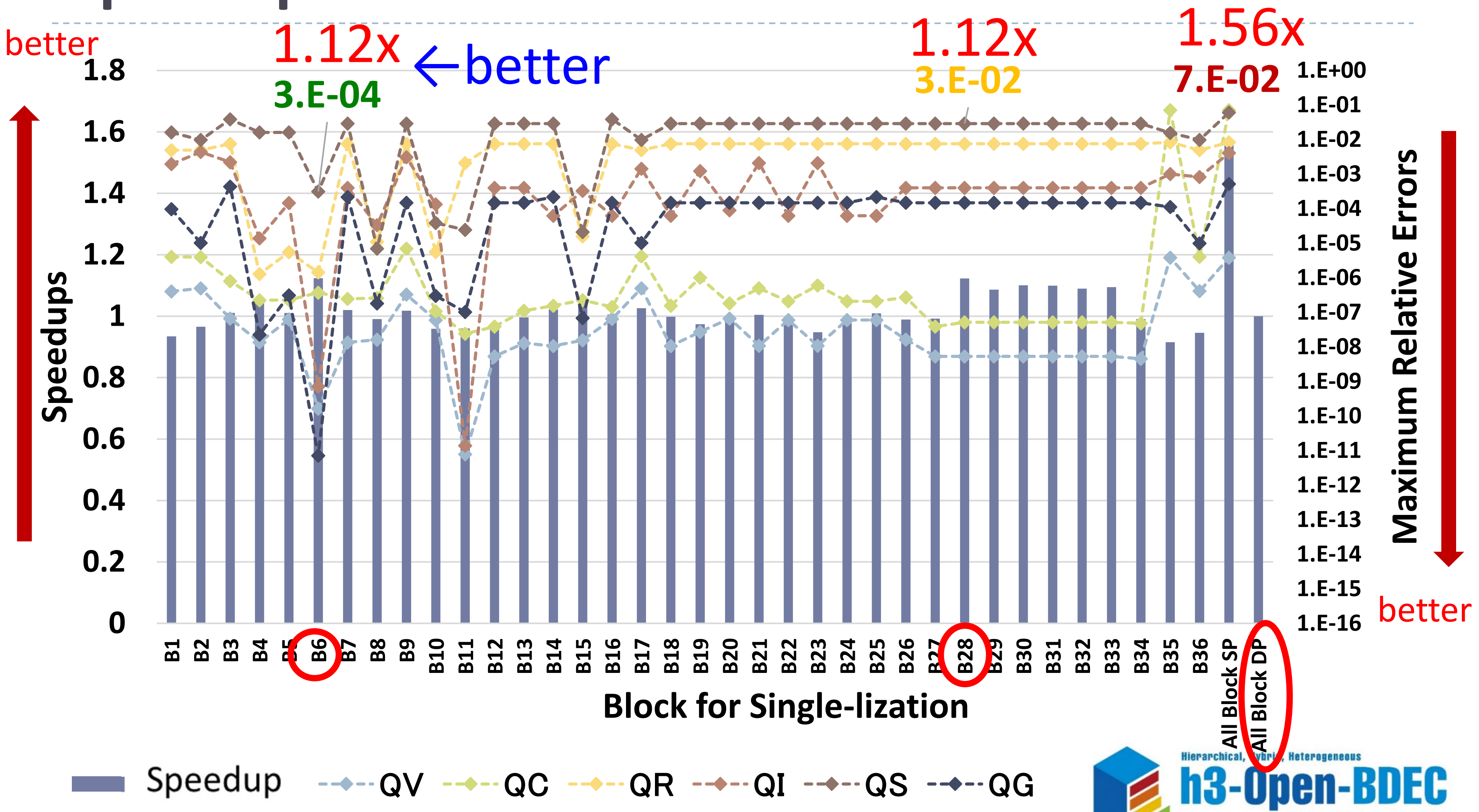
Result of 2. Blocks: Speedup and Energy Reduction



■ Speedup ■ Energy Efficiency

Hierarchical, Hybrid, Heterogeneous h3-Open-BDEC Big Data & Extreme Computing
名古屋大学 NAGOYA UNIVERSITY

Result of 2. Blocks: Speedup and Maximum Relative Errors



1.12x
3.E-04
←better

1.12x
3.E-02

1.56x
7.E-02

Checksum Arrays (6 Kinds)

Outline

- TOPIC I
  - Mixed-Precision and Energy Optimization by ppOpen-AT
- TOPIC II
  - Explainable AI for Auto-tuning on an Accurate Precision Matrix-Matrix Multiplication Library

Collaboration with
Mr. Shota Aoki, Master Course Student,
Graduate School of Informatics, Nagoya University

Explainable AI for Auto-tuning of Numerical Libraries

Background

- Several social problems occur due to usage of results from artificial intelligence (AI) without verification.
- Human check is needed from output from AI.
- To reduce cost of tuning process, several technologies of software auto-tuning (AT) is developing.
  - Currently, AI is applied for the AT function.

→ In this study, we verify AI result for performance tuning

of a numerical library for accuracy assurance
to show "explainability of AI output."

Explainable AI（XAI）

- Can we explain output from AI?
  - Explainable AI (XAI)
- Explainable AI is classified as follows.[II-1]
  - Explainability
    - Technology for explanation of predicted results to help human understanding.
    - Ex) Tools for LIME, SHAP
  - Interpretability
    - Technology for understanding of computation process up to final prediction by analyzing inside data structures.
    - Ex) Making decision tree.

→In this study, we focus on Explainability.

[II-1] 大坪ほか：「XAI(説明可能なAI)：そのとき人工知能はどう考えたのか」、リックテレコム、2021

SHAP（SHapley Additive exPlanations） [II-2]

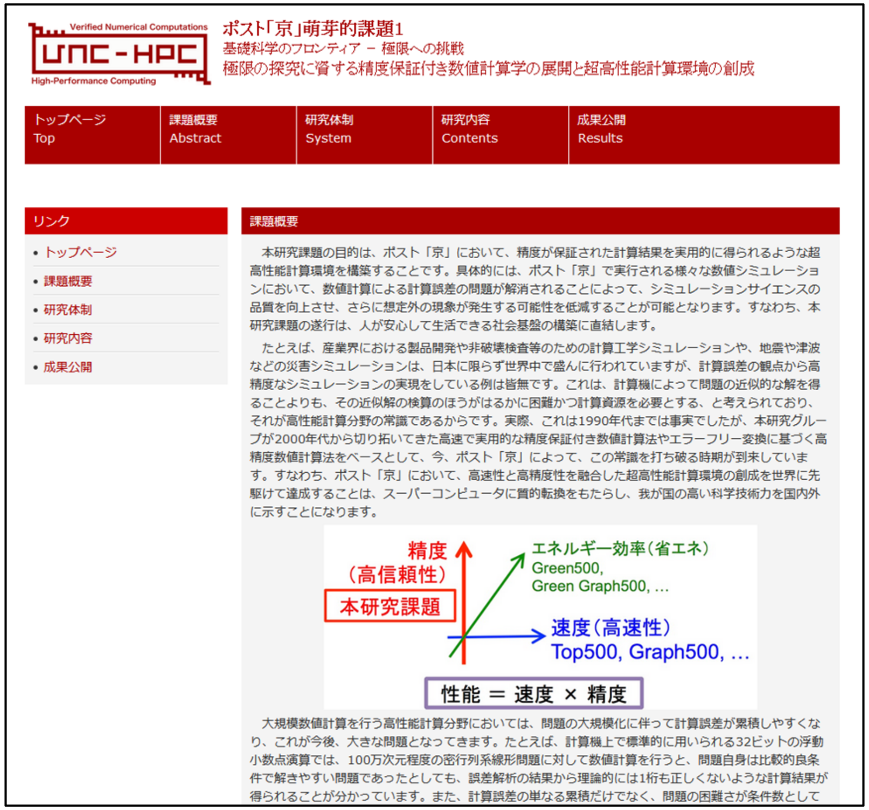
- Shapley Value for collaborative game theory is applied.
  - There is validity.
- Approximately Shapley value is calculated.
  - Ensemble model for tree : High speed and accurate Shapley value.
  - Deep learning model : High speed and approximate Shapley value.
  - General algorithms : Estimated Shapley value.
- Drawback
  1. High Computation Complexity
     →Approximate value is only solution to use.

[II-2] S. Lundberg, S-I. Lee, A Unified Approach to Interpreting Model Predictions, 2017
https://arxiv.org/abs/1705.07874

VNC-HPC Library

- Post-K Exploratory Research1: 基礎科学のフロンティアー極限への挑戦ー:「極限の探究に資する精度保証付き数値計算学の展開と超高性能計算環境の創成」 (PI : Prof. Takeshi Ogita at Tokyo Women’s Cristian University, ~2019)
- Environment of high performance computing is developing to maintain computation accuracy.
- The following library is released as open source software.

1. **OzBLAS: Accurate and Reproducible BLAS based on Ozaki scheme** [for PC, GPU]
2. **GEMMTC: GEMM using Tensor Cores** [for GPU]
3. **DHPMM_F for GPU: High-precision Matrix Multiplication with Faithful Rounding** [for GPU]
4. **PDDOTK: K-fold Precision Dot Product** [for PC, FX100]
5. **BLAS-DOT2: Higher-precision BLAS based on Dot2** [for GPU]
6. **LINSYS_VR: Verified Solution of Linear Systems with Directed Rounding** [for K Computer, FX100]
7. **LINSYS_V: Verified Solution of Linear Systems** [for PC, K Computer, FX100]
8. **DHPMM_F: High-precision Matrix Multiplication with Faithful Rounding** [for PC, K Computer, FX100]



ポスト「京」萌芽的課題1
基礎科学のフロンティア － 極限への挑戦
極限の探究に資する精度保証付き数値計算学の展開と超高性能計算環境の創成

課題概要

本研究課題の目的は、ポスト「京」において、精度が保証された計算結果を実用的に得られるような超高性能計算環境を構築することです。具体的には、ポスト「京」で実行される様々な数値シミュレーションにおいて、数値計算による計算誤差の問題が解消されることによって、シミュレーションサイエンスの品質を向上させ、さらに想定外の現象が発生する可能性を低減することが可能となります。すなわち、本研究課題の遂行は、人が安心して生活できる社会基盤の構築に直結します。

たとえば、産業界における製品開発や非破壊検査等のための計算工学シミュレーションや、地震や津波などの災害シミュレーションは、日本に限らず世界中で盛んに行われていますが、計算誤差の観点から高精度なシミュレーションの実現をしている例は皆無です。これは、計算機によって問題の近似的な解を得ることよりも、その近似解の検算のほうがはるかに困難かつ計算資源を必要とする、と考えられており、それが高性能計算分野の常識であるからです。実際、これは1990年代までは事実でしたが、本研究グループが2000年代から切り拓いてきた高速で実用的な精度保証付き数値計算法やエラーフリー変換に基づく高精度数値計算法をベースとして、今、ポスト「京」によって、この常識を打ち破る時期が到来しています。すなわち、ポスト「京」において、高速性と高精度性を融合した超高性能計算環境の創成を世界に先駆けて達成することは、スーパーコンピュータに質的転換をもたらし、我が国の高い科学技術力を国内外に示すことになります。

性能 ＝ 速度 × 精度

大規模数値計算を行う高性能計算分野においては、問題の大規模化に伴って計算誤差が累積しやすくなり、これが今後、大きな問題となってきます。たとえば、計算機上で標準的に用いられる32ビットの浮動小数点演算では、100万次元程度の密行列系線形問題に対して数値計算を行うと、問題自身は比較的良条件で解きやすい問題であったとしても、誤差解析の結果から理論的には1桁も正しくないような計算結果が得られることが分かっています。また、計算誤差の単なる累積だけでなく、問題の困難さが条件数として

HP : http://www.math.twcu.ac.jp/ogita/post-k/index.html

Background, objective, and motivation of our work

- High-accuracy and low-accuracy calculations are one of the important calculation techniques which can solve large-scale and complicated calculations.
- Some studies have investigated the accuracy assurance of BLAS and LAPACK.
  - These studies have used mixed procedure computations and arbitrary digit computations.
  - Most of BLAS and LAPACK libraries tend to place less emphasis on the accuracy of the computation results.
- Why high-accuracy and assured calculations are not widespread?
- => TIME
- We consider to calculate it on GPU and shorten the time.
- This work will be helpful for large-scale and complicated calculations on current and future generation computers. (⇨ topics of interest of this workshop)

In particular, we will focus on the following topics of interest, but not limited to:

- Programming models, languages and frameworks for facilitating HPC software evolution and refactoring.
- Algorithms and implementation methodologies for future-generation computing systems, including manycores and accelerators (GPUs, Xeon Phi, etc).
- Automatic performance tuning techniques, runtime systems and domain-specific languages for hiding the complexity of underlying system architectures.
- Practices and experiences on porting of legacy applications and libraries.

High-precision matrix-matrix multiplication algorithm

- Our target calculation: assured matrix-matrix multiplication (MMM) method proposed by Ozaki et al.
  - hereinafter, we refer to this MMM method as the Ozaki method
- Overview of the Ozaki method:

consider $C = AB$

- A : a matrix of size m * l
- B : a matrix of size l * n
- C : a matrix of size m * n

Step1: error-free transformation

$$A = A^{(1)}+A^{(2)}+A^{(3)}+\ldots+A^{(p)}$$
$$B = B^{(1)}+B^{(2)}+B^{(3)}+\ldots+B^{(q)}$$

The elements in the matrices with lower indices are given with a higher number of digits.

Step2: individual MMM

$$\begin{aligned}AB&=(A^{(1)}+A^{(2)}+\ldots+A^{(p)})(B^{(1)}+B^{(2)}+\ldots+B^{(q)})\\&=\underline{A^{(1)}\,B^{(1)}}+\underline{A^{(1)}\,B^{(2)}}+\underline{A^{(2)}\,B^{(1)}}+\ldots+\underline{A^{(p)}\,B^{(q)}}\end{aligned}$$

Step3: Accurate Sum

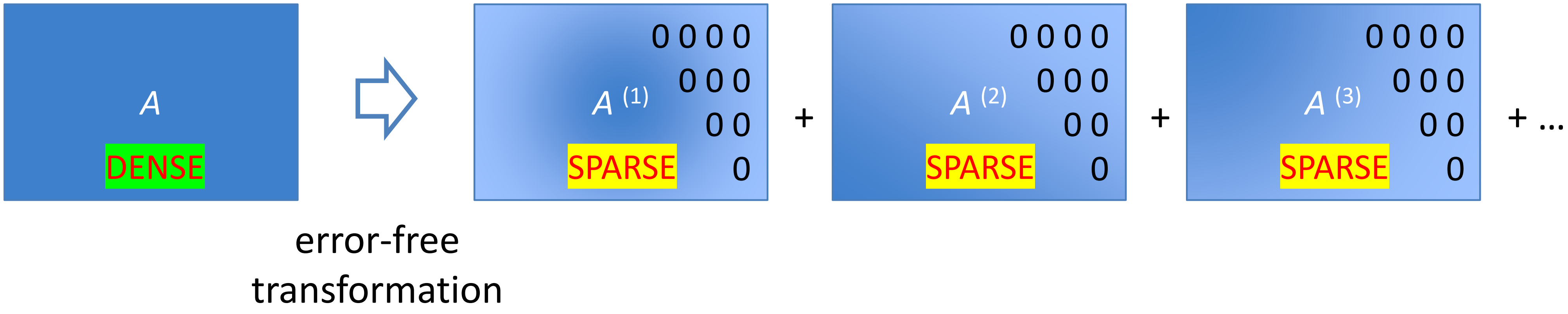
$fl(A^{(i)}\,B^{(j)}\,)=A^{(i)}\,B^{(j)}$ for $1 \leqq i \leqq p,\ 1 \leqq j \leqq q$.

$$\begin{aligned}&= fl(A^{(1)}\,B^{(1)}\,) + fl(A^{(1)}\,B^{(2)}\,) + fl(A^{(2)}B^{(1)}\,) + \ldots\\&+ fl(A^{(p)}\,B^{(q)}\,)\\&= C_1 + C_2 + \ldots + C_{pq}\end{aligned}$$

fl is a floating-point arithmetic with rounding to the nearest

Previous work and Proposed method

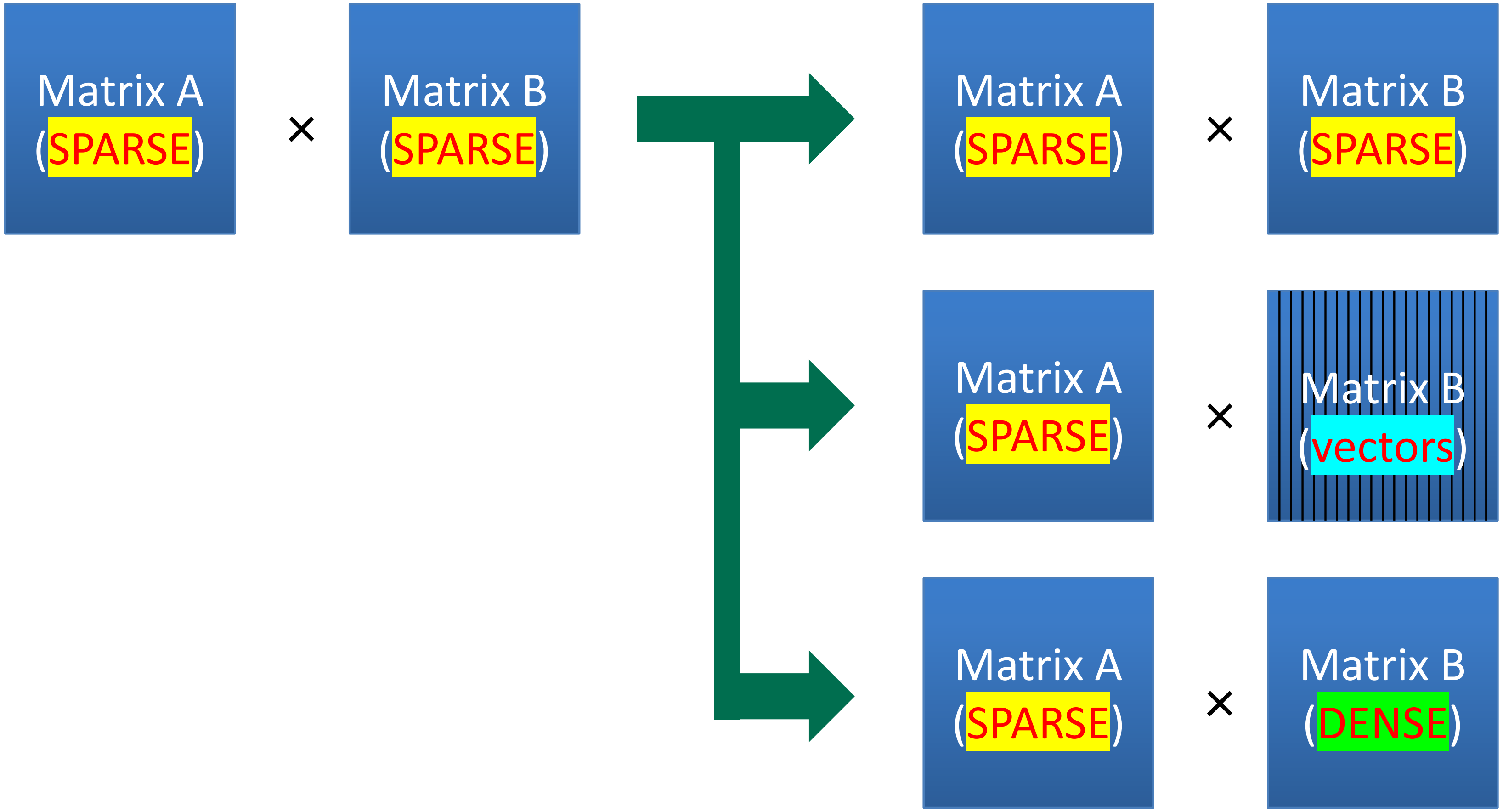
- When the value ranges of the input matrix elements are large, error-free transformation generates many sparse matrices.



- In this case, many double precision general matrix - matrix multiplication (dgemm) are performed. Transforming dense matrices into sparse matrices and performing sparse matrix operations will require shorter calculation time than dense matrix operations.

- Therefore, our previous work proposed to transform the target matrices into sparse matrices and calculate sparse matrix computations on CPU.
  - Considering the performance, sparse matrix - vector multiplications (SpMV) are used.
- In this study, we propose to calculate these sparse matrix computations on GPU.

How to calculate Sparse Matrix A - Matrix B multiplication?



Implementation Details of Accurate MMM Library (Ozaki Method)

1. Implementation by dgemm 1dgemm
2. SpMV (Inner Parallel) with CRS format 2CRS
3. SpMV (Outer Parallel) with CRS format 3CRS
4. SpMV (Multiple RHS, Inner Parallel) with CRS format 4CRS
5. SpMV (Multiple RHS, Inner Parallel (Blocking)) with CRS format 5CRS
6. SpMV (Inner Parallel) with ELL format 6ELL
7. SpMV (Outer Parallel) with ELL format 7ELL
8. SpMV (Multiple RHS, Inner Parallel) with ELL format 8ELL
9. SpMV (Multiple RHS, Inner Parallel (Blocking)) 9ELL
10. Implementation by Batched BLAS 10GPU
11. Implementation by dgemm (GPU) 11GPU
12. SpMV with CRS format (GPU) 12GPU
13. SpMV with ELL format (GPU) 13GPU
14. SpMM with CRS format (GPU) 14GPU

Experimental Environment

- **Supercomputer “Flow” TypeII Subsystem**
  Information Technology Center, Nagoya University
  - CPU
    - Intel Xeon Gold 6230, 20 Cores, 2.10 - 3.90 GHz x 2 Sockets
  - GPU : Model generation for machine learning.
    - NVIDIA Tesla V100 (Volta) SXM2, 2,560 FP64 Cores, up to 1,530 MHz x 4 Sockets
- LIME : ver. 0.2.0.1 (In this presentation, we skip this results.)
- SHAP: ver.0.39.0
- Model of Machine Learning (Classifier)
  - Random Forest Model by scikit-learn ver. 0.24.1
  - 11 Kinds of Implementations are used.

How to generate test matrices

- Random matrices
- Matrix #1 : Elements are generated by 0-1 region for matrix A and B, then insert a value with **pow(10,rand( )%Φ)** with respect to a sparsity.
  - Up to Φ=30.
    (If Φ is too large, then it cannot treat it by python.)
- Matrix #2: Elements are generated by 0-1 region for matrix A and B with respect to a sparsity.
  - The sparsity is from **90% to 98%** in this experience.
- Matrix #3 : Elements are generated by Identity Matrix for matrix A and B, then insert a value with **pow(10,rand( )%Φ)** with respect to a sparsity.
  - Up to Φ=30.
- The random seed is fixed.

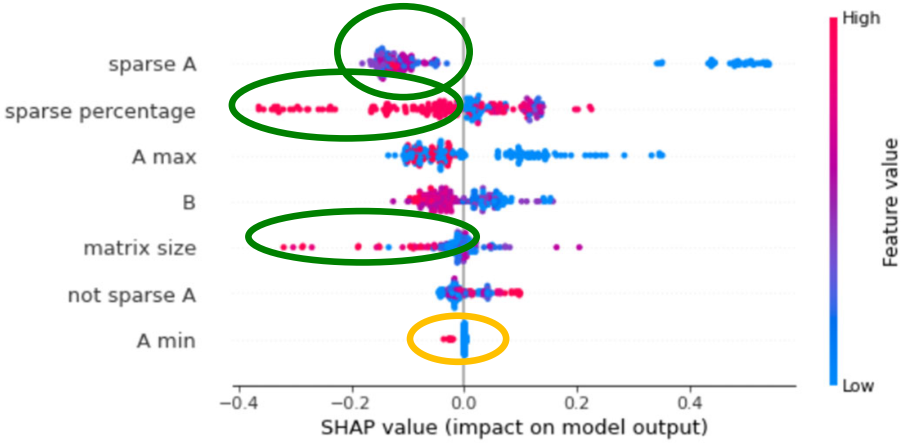
Number of Learning Data and Prediction Accuracy

- Learning Data
  1. Matrix #1： It can generate huge value of elements. The sparsity of almost 0.
  2. Matrix #2： Sparse matrices with 0-1 range of elements.
  3. Matrix #3：： It can generate huge value of elements. It can generate arbitrary sparsity.
  - Matrix size: 1000 – 4000.
- Information for Machine Learning
  - Model：Random forest
  - Explainable valuables：7 variables
    - 1)Matrix size; 2)Sparsity of input matrix; 3)Maximum element of A; 4)Minimum element of A; 5)Number of split for sparse of A ; 6)Number of Split for dense of A; 7)Number of split for B
  - Number of learning (training) data: 199
  - Number of test data: 23
  - Accuracy: 91.3%

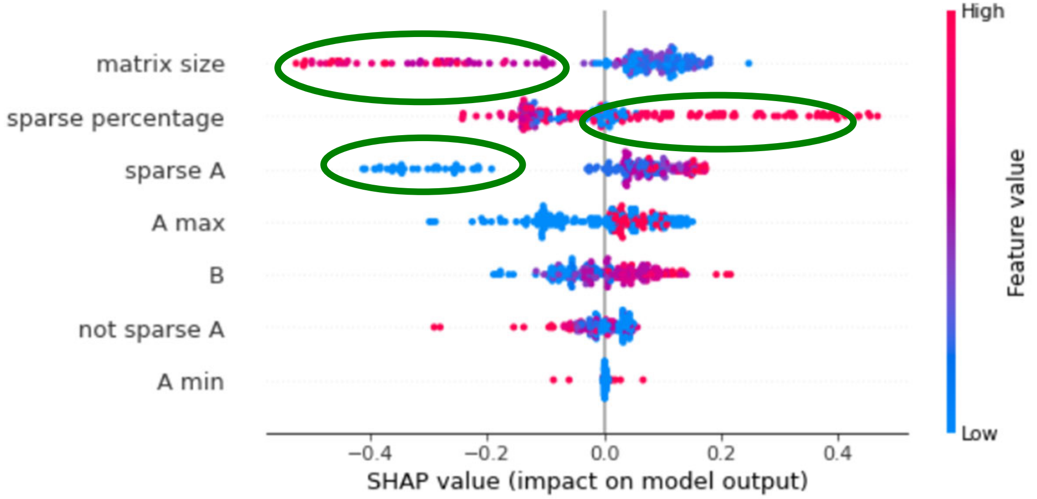
Analysis Result by SHAP

- **Remarkable factors: (1)Absolute SHAP Value is large (Crucial Factor) , (2) Same color and close (Same value of explainable variables) , (3)Large cluster, or form a queue (Number of cases is large.)**

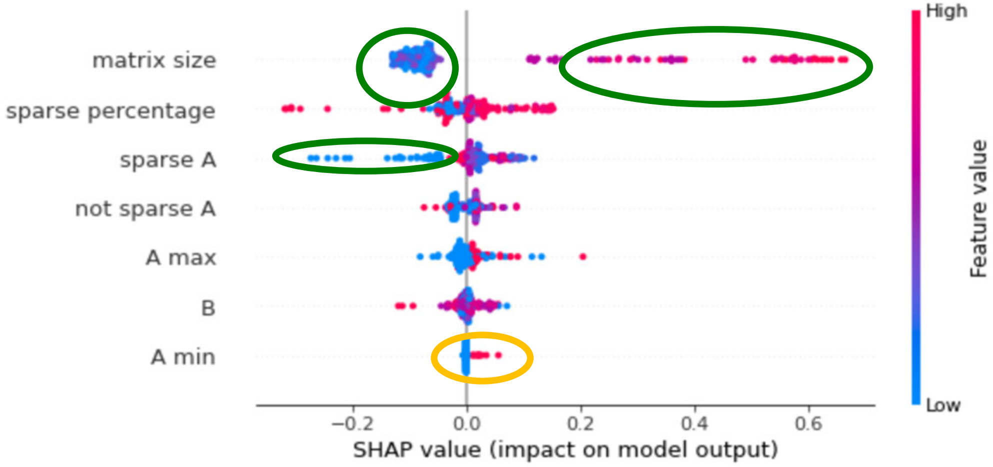
(1) Select: dgemm



(2) Select: SpMV with CRS format (Multiple RHS and Inner Parallel)



(3)Select: SpMM with CRS format (GPU)



Finding crucial factors:

- (1) **dgemm** : Number of splits for sparse of A (-), Sparsity of input matrix (-), Matrix sizes (-) →Reasonable
- (2) **CRS (SpMV)** : Matrix sizes (-), Sparsity of input matrix (+), Number of splits for sparse of A (-) →Reasonable
- (3) **CRS (SpMM)(GPU)** : Matrix sizes (-, +), Number of splits for sparse of A (-)
- Almost all element of minimal value for A is 0. **→No effect. This is a NG explainable variable.**

Closing Remarks

- TOPIC I: A Proposal of Mixed-Precision and/or Energy Optimization for ppOpen-AT.
  - Proposed an AT framework (directive) for optimization of mixed-precision computations and/or energy
  - Future work
    - Evaluate several applications
    - Supply the system **as a tool** : It is useful to show history by changing target of parts in programs (variables/arrays, or blocks).
- TOPIC II: Explainable AI for Auto-tuning of Numerical Libraries.
  - Obtained a case with reasonable explanation on a numerical library.
  - Future work: Propose AT method **to reduce AT time**, or **select better expandable variables** automatically.

Hierarchical, Hybrid, Heterogeneous h3-Open-BDEC Big Data & Extreme Computing

名古屋大学 NAGOYA UNIVERSITY