Auto-tuning for Computation Accuracy and Power Consumption by ppOpen-AT

Takahiro Katagiri (Information Technology Center, Nagoya University, E-mail: katagiri@cc.nagoya-u.ac.jp / RIKEN R-CCS, Visiting Researcher)

ppOpen-AT: An Auto-tuning (AT) Language

- An AT language to add AT function to arbitrary programs.
- By adapting a dedicated preprocessor, the followings are generated:
  - Multiple candidates to optimize the target program.
  - Code with searching function of performance parameters.

```
!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
```

Example of loop unrolling with depth from 1st to 4th .

Generate the code by dedicated preprocessor of ppOpen-AT

```
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
```

Generated loop unrolling codes with 3rd depth.

T. Katagiri, S. Ohshima, M. Matsumoto, Auto-tuning of computation kernels from an FDM Code with ppOpen-AT, 2014 IEEE 8th International Symposium on Embedded Multicore/Manycore SoCs, pp,91-98 (2014)

Specifying Optimization for Mixed-Precision Computations

An Example) Block Level

```
!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
```

A Sample Program (Directives for Blocks.)

Substitutions from DP to SP, and from SP to DP, are generated.

Generate the code by preprocessor.

```
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(:,:)
```

Replace valuables / arrays /constant in the target block to SP.

Replace to SP.

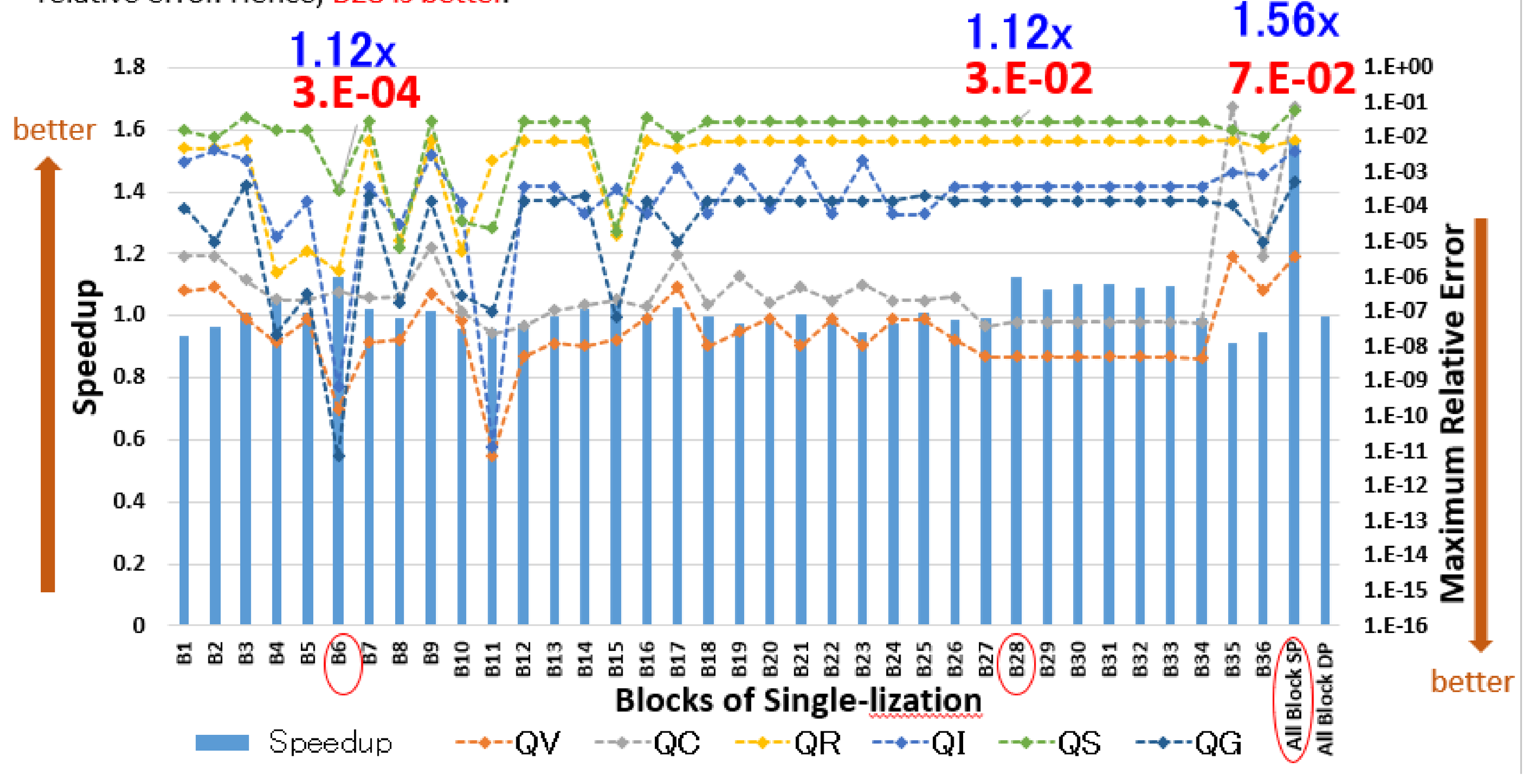
An Example for Generated Code(Single-lization for the Block 1

Results

- **The Supercomputer "Flow" Type I** Subsystem ("Fugaku" Type), ITC, Nagoya University.

② **Blocks:** Comparison with Speedups and Maximum Relative Errors in each group (Single−lization)

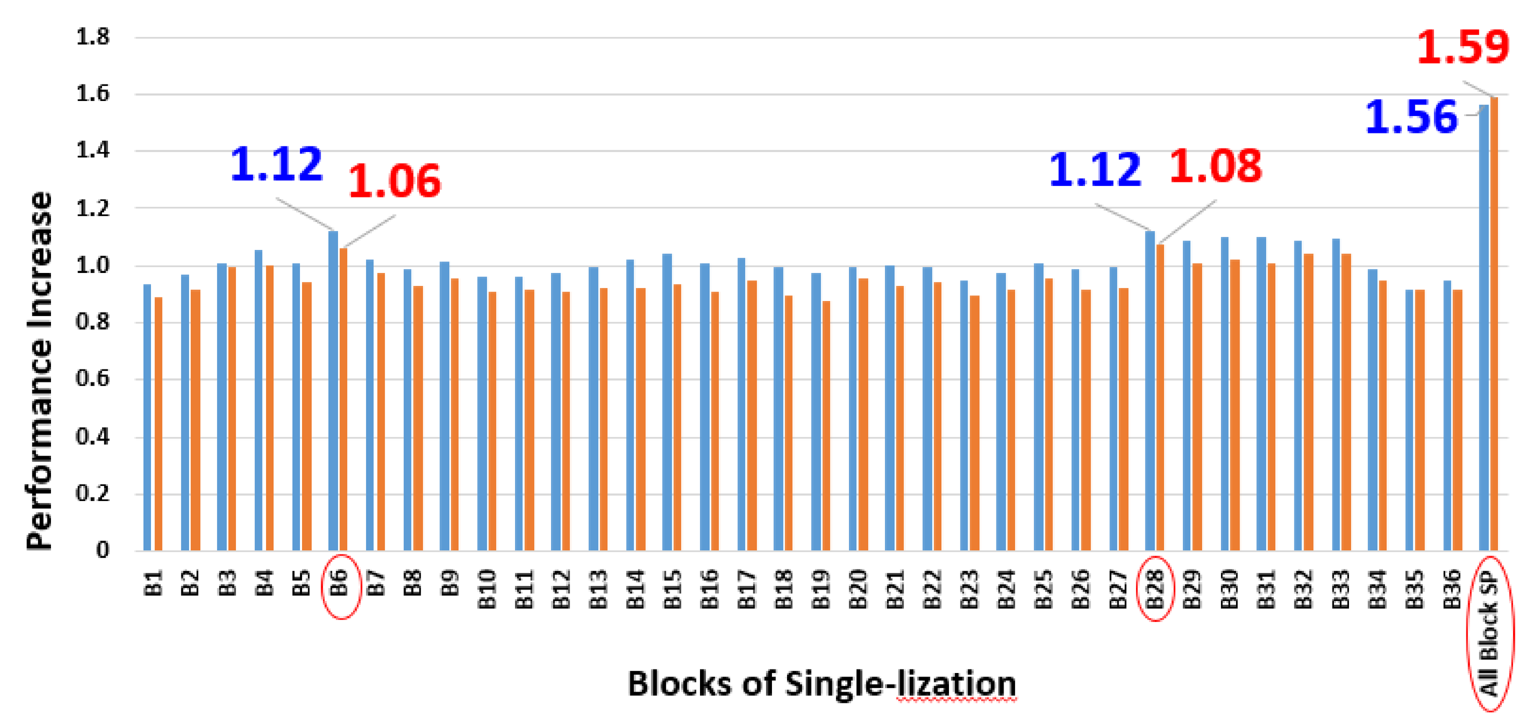
- In All Block SP, all of output in NICAM extends 1.E-07 in maximum relative error. The maximum one is 7.E-02 in QC.
- In B6 and B28, their speedups are 1.12 times, but 3.E-02 in B28, while B6 is 3.E-04 in B6 for maximum relative error. Hence, B28 is better.



- NICAM : Global Cloud Resolving Model
- Nicam_dckernel_2016: One of benchmark packages.
  - A very long-body, three-nested loop.

②**Blocks:** Speedup and Reduction ratio of Energy in each group (Single−lization)

- There is no different tendency between speedup and reduction of energy consumption.
- The best reduction of energy consumption is All Block SP; it establishes 1.59 times.
- The second is B28; it establishes 1.08 times, and the third is B6; it establishes 1.06 times, and so on.



Information Technology Center, Nagoya University
http://www.icts.nagoya-u.ac.jp/en/center/