Optimizing Multigrid Poisson Solver of Cartesian CFD code CUBE

Kazuto Ando1,2, Rahul Bale2, Keiji Onishi2, Kiyoshi Kumahata1, Kazuo Minami1, and Makoto Tsubokura2,3

1 Operations and Computer Technologies Division, RIKEN Center for Computational Science (R-CCS), Japan
2 Complex Phenomena Unified Simulation Research Team, RIKEN Center for Computational Science (R-CCS), Japan
3 Department of Computational Science, Graduate School of System Informatics, Kobe University, Japan

Introduction

Evaluation of cost distribution
• The cost per time-step is dominated by the Poisson solver which accounts for 91%.
• Specifically, the Gauss-Seidel method based smoother accounts for 73% of the cost.
• The Gauss-Seidel loop included in this smoother is the highest cost component of the time-step loop (50%).
• Routine for local halo exchange between inner cubes is called from many parts in program and its cost is 26% of the time-step loop.

Optimization of computational kernel (Gauss-Seidel loop)

Byte per Flop of Original Code
Considering 32 iterations
• Number of floating point operations
  (5 add + 1 subtract + 2 multiply) x 32 iterations = 256 operations
• Memory access
  128 Byte x (15 x 2 + 3) lines = 4224 Byte
• Byte per Flop
  4224 Byte / 256 operations = 16.5
• Attainable peak performance ratio
  0.36 / 16.5 x 100 = 2.18%

Byte per Flop of Optimized Code
Considering 32 iterations
• Number of floating point operations
  (5 add + 1 subtract + 2 multiply) x 32 iterations = 256 operations
• Memory access
  128 Byte x 3 lines x (2 load + 1 store x 2) = 1536 Byte
• Byte per Flop
  1536 Byte / 256 operations = 6
• Attainable peak performance ratio
  0.36 / 6 x 100 = 6%

Optimization Result
• Using pointer to directly refer to the array “qcnt” decreases integer operations.
• Apply swapping array dimensions and dividing array elements for “Black” cells and “Red” cells of Red-Black ordering.
• These care for stride access in first dimension of “qcnt” reduces wait time due to floating point load memory access.
• This result in a 9.8x speed up, and the peak ratio increases from 0.4% to 4.0%

Conclusion
• We optimized multigrid Poisson solver which dominates 91% of execution time of the main time-stepping loop.
• With optimization of the computational kernel (Gauss-Seidel loop), we achieved a 9.8x speedup and peak ratio increased from 0.4% to 4.0%
• With optimization of parallel performance, we achieved sustainable 2.9x–3.9x speedup up to 8,192 nodes in time-step loop.
• In case of using more number of nodes, scaling of CG method loop gets worse, therefore more MPI process level tuning is required.