Multigrid Method using OpenMP/MPI Hybrid Parallel Programming Model on Fujitsu FX10

Kengo Nakajima
Information Technology Center, The University of Tokyo, Japan

November 14th, 2012
Fujitsu Booth SC12
Salt Lake City, Utah, USA
Motivation of This Study

• Parallel Multigrid Solvers for FVM-type appl. on Fujitsu PRIMEHPC FX10 at University of Tokyo (Oakleaf-FX)

• Flat MPI vs. Hybrid (OpenMP+MPI)

• Expectations for Hybrid Parallel Programming Model
  – Number of MPI processes (and sub-domains) to be reduced
  – $O(10^8-10^9)$-way MPI might not scale in Exascale Systems
  – Easily extended to Heterogeneous Architectures
    • CPU+GPU, CPU+Manycores (e.g. Intel MIC/Xeon Phi)
    • MPI+X: OpenMP, OpenACC, CUDA, OpenCL
Multigrid

• Scalable Multi-Level Method using Multilevel Grid for Solving Linear Eqn’s
  – Computation Time ~ O(N) (N: # unknowns)
  – Good for large-scale problems

• Preconditioner for Krylov Iterative Linear Solvers
  – MGCG
Around the multigrid in a single slide

- Multigrid is a scalable method for solving linear equations.
- Relaxation methods (smoother/smoothing operator in MG world) such as Gauss-Seidel efficiently damp high-frequency error but do not eliminate low-frequency error.
- The multigrid approach was developed in recognition that this low-frequency error can be accurately and efficiently solved on a coarser grid.
- Multigrid method uniformly damps all frequencies of error components with a computational cost that depends only linearly on the problem size (=scalable).
  - Good for large-scale computations
- Multigrid is also a good preconditioning algorithm for Krylov iterative solvers.
Multigrid is scalable
Weak Scaling: Problem Size/Core Fixed
for 3D Poisson Eqn’s ($\Delta\phi = q$)
MGCG = Conjugate Gradient with Multigrid Preconditioning

![Graphs showing iterations and seconds versus DOF for different methods.](image)
Multigrid is scalable
Weak Scaling: Problem Size/Core Fixed
Comp. time of MGCG for weak scaling is constant:
=> scalable
Flat MPI vs. Hybrid

Flat-MPI: Each PE -> Independent

Hybrid: Hierarchal Structure
Current Supercomputer Systems
University of Tokyo

- Total number of users ~ 2,000
  - Earth Science, Material Science, Engineering etc.
- Hitachi HA8000 Cluster System (T2K/Tokyo) (2008.6-)
  - Cluster based on AMD Quad-Core Opteron (Barcelona)
  - Peak: 140.1 TFLOPS
- Hitachi SR16000/M1 (Yayoi) (2011.10-)
  - Power 7 based SMP with 200 GB/node
  - Peak: 54.9 TFLOPS
- Fujitsu PRIMEHPC FX10 (Oakleaf-FX) (2012.04-)
  - SPARC64 IXfx
  - Commercial version of K computer
  - Peak: 1.13 PFLOPS (1.043 PF, 21st, 40th TOP 500 in 2012 Nov.)
• Aggregate memory bandwidth: 398 TB/sec.
• Local file system for staging with 1.1 PB of capacity and 131 GB/sec of aggregate I/O performance (for staging)
• Shared file system for storing data with 2.1 PB and 136 GB/sec.
• External file system: 3.6 PB
Target Application

• 3D Groundwater Flow via. Heterogeneous Porous Media
  – Poisson’s equation
  – Randomly distributed water conductivity
  – Distribution of water conductivity is defined through methods in geostatistics [Deutsch & Journel, 1998]

• Finite-Volume Method on Cubic Voxel Mesh

• Distribution of Water Conductivity
  – $10^{-5}$-$10^{+5}$, Condition Number $\sim 10^{+10}$
  – Average: 1.0

• Cyclic Distribution: $128^3$
Linear Solvers

• Preconditioned CG Method
  – Multigrid Preconditioning (MGCG)
  – IC(0) for Smoothing Operator (Smoother): good for ill-conditioned problems

• Parallel Geometric Multigrid Method
  – 8 fine meshes (children) form 1 coarse mesh (parent) in isotropic manner (octree)
  – V-cycle
  – Domain-Decomposition-based: Localized Block-Jacobi, Overlapped Additive Schwartz Domain Decomposition (ASDD)
  – Operations using a single core at the coarsest level (redundant)
Overlapped Additive Schwartz
Domain Decomposition Method

ASDD: Localized Block-Jacobi Precond. is stabilized

Global Operation

$$Mz = r$$

Local Operation

$$z_{\Omega_1} = M_{\Omega_1}^{-1} r_{\Omega_1}, \quad z_{\Omega_2} = M_{\Omega_2}^{-1} r_{\Omega_2}$$

Global Nesting Correction

$$z_{\Omega_1}^{n} = z_{\Omega_1}^{n-1} + M_{\Omega_1}^{-1} (r_{\Omega_1} - M_{\Omega_1} z_{\Omega_1}^{n-1} - M_{\Gamma_1} z_{\Gamma_1}^{n-1})$$

$$z_{\Omega_2}^{n} = z_{\Omega_2}^{n-1} + M_{\Omega_2}^{-1} (r_{\Omega_2} - M_{\Omega_2} z_{\Omega_2}^{n-1} - M_{\Gamma_2} z_{\Gamma_2}^{n-1})$$
Computations on Fujitsu FX10

- Fujitsu PRIMEHPC FX10 at U.Tokyo (Oakleaf-FX)
  - 16 cores/node, flat/uniform access to memory
- Up to 4,096 nodes (65,536 cores) (Large-Scale HPC Challenge)
  - Max 17,179,869,184 unknowns
  - Flat MPI, HB 4x4, HB 8x2, HB 16x1
    - HB MxN: M-threads x N-MPI-processes on each node
- Weak Scaling
  - $64^3$ cells/core
- Strong Scaling
  - $128^3 \times 8 = 16,777,216$ unknowns, from 8 to 4,096 nodes
- Network Topology is not specified
  - 1D
<table>
<thead>
<tr>
<th>L1</th>
<th>L1</th>
<th>L1</th>
<th>L1</th>
<th>L1</th>
<th>L1</th>
<th>L1</th>
<th>L1</th>
<th>L1</th>
<th>L1</th>
<th>L1</th>
<th>L1</th>
<th>L1</th>
<th>L1</th>
<th>L1</th>
<th>L1</th>
<th>L1</th>
<th>L1</th>
<th>L1</th>
<th>L1</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
</tr>
</tbody>
</table>

**HB**  \( M \times N \)

- **Number of OpenMP threads per a single MPI process**
- **Number of MPI processes per a single node**
Coarse Grid Solver on a Single Core

Original Approach

Size of the Coarsest Grid = Number of MPI Processes
Redundant Process

In Flat-MPI, this size is larger
Weak Scaling: up to 4,096 nodes
up to 17,179,869,184 meshes (64^3 meshes/core)
Although ASDD is applied, convergence is getting worse for larger number of nodes/domains, **DOWN is GOOD**
Strategy: Coarse Grid Aggregation

- Decreasing number of MPI processes at coarser level.
- Switching to redundant processes for coarse grid solvers earlier (i.e. at finer level).
  - Node-to-node communications at coarser levels are reduced.
- Coarse grid solver on a single MPI proc., not a single core
  - HB 4x4: 4 cores
  - HB 8x2: 8 cores
  - HB 16x1: 16 cores, Single Node
    - Info. gathered to a single MPI process
    - OpenMP is needed
- In post-peta/exa-scale systems, each node will consists $O(10^2)$ of cores, therefore utilization of these many cores on each node should be considered.
Coarse Grid Aggregation: at lev=2

PE#0

PE#1

PE#2

PE#3

lev=1 lev=2 lev=3 lev=4
Coarse Grid Aggregation: at lev=2

Apply multigrid procedure on a single MPI process.

Trade-off: Coarse grid solver is more expensive than original approach.
Results at 4,096 nodes

lev: switching level to “coarse grid solver”

Opt. Level = 7

DOWN is GOOD
Weak Scaling: up to 4,096 nodes
up to 17,179,869,184 meshes (64³ meshes/core)
Convergence has been much improved by coarse grid aggregation, **DOWN is GOOD**
Strong Scaling: up to 4,096 nodes
268,435,456 meshes, only $16^3$ meshes/core at 4,096 nodes

UP is GOOD
Strong Scaling at 4,096 nodes
268,435,456 meshes, only $16^3$ meshes/core at 4,096 nodes

<table>
<thead>
<tr>
<th>Iterations</th>
<th>58</th>
<th>49</th>
<th>63</th>
<th>51</th>
<th>63</th>
<th>51</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parallel performance (%)</td>
<td>2.97</td>
<td>13.6</td>
<td>5.72</td>
<td>16.2</td>
<td>8.25</td>
<td>19.0</td>
</tr>
</tbody>
</table>
Summary

• “Coarse Grid Aggregation” is effective for stabilization of convergence at $O(10^4)$ cores for MGCG
  – Not so effective on communications
  – HB 8x2 is the best at 4,096 nodes
    • HB programming model with smaller number of MPI processes (e.g. HB 8x2, HB 16x1) are better, if the number of nodes are larger.
      – Smaller problem size for coarse grid solver
      – If the number of nodes are larger, performance is better

• Further Optimization/Tuning
  – Single node/core performance for FX10
    • current code is optimized for T2K/Tokyo (cc-NUMA)
  – Overlapping of computation & communication
    • more difficult than SpMV
    – Automatic selection of the optimum switching level $lev$
  – Gradual reduction of MPI process number (e.g. 8192-512-32-1)
Reference:

Kengo Nakajima
“OpenMP/MPI Hybrid Parallel Multigrid Method on Fujitsu FX10 Supercomputer System”