ppOpen-HPC: Open Source Infrastructure for Development and Execution of Large-Scale Scientific Applications on Post-Peta Scale Supercomputers with Automatic Tuning (AT)

Kengo Nakajima
Information Technology Center, The University of Tokyo

San Diego Supercomputer Center, La Jolla, CA, USA
July 30, 2015
Summary

• ppOpen-HPC is an open source infrastructure for development and execution of optimized and reliable simulation code on post-peta-scale (pp) parallel computers based on many-core architectures with automatic tuning (AT), and it consists of various types of libraries, which cover general procedures for scientific computation.

• Some Recent Activities
  – Recent achievements in the development of ppOpen-MATH/MG, which is a geometric multigrid solver
  – Coupled earthquake simulations by ppOpen-MATH/MP
  – An example of AT by ppOpen-AT on 3D FDM code of seismic simulations (Seism3D)
Lessons learned in the 20\textsuperscript{th} Century

• Methods for scientific computing (e.g. FEM, FDM, BEM etc.) consists of typical data structures, and typical procedures.
  • Optimization of each procedure is possible and effective.

• Well-defined data structure can “hide” communication processes with MPI from code developer.
  • Code developers do not have to care about communications
  • Halo for parallel FEM
• ppOpen-HPC
• ppOpen-MATH
  – ppOpen-MATH/MP: Coupler
• ppOpen-AT
ppOpen-HPC: Overview

- Application framework with automatic tuning (AT)
  - “pp” : post-peta-scale
- Five-year project (FY.2011-2015) (since April 2011)
  - P.I.: Kengo Nakajima (ITC, The University of Tokyo)
  - Part of “Development of System Software Technologies for Post-Peta Scale High Performance Computing” funded by JST/CREST (Supervisor: Prof. Mitsuhisa Sato, Co-Director, RIKEN AICS)

- Team with 7 institutes, >50 people (5 PDs) from various fields: Co-Design
  - ITC/U.Tokyo, AORI/U.Tokyo, ERI/U.Tokyo, FS/U.Tokyo
  - Hokkaido U., Kyoto U., JAMSTEC
• **Group Leaders**
  – Masaki Satoh (AORI/U.Tokyo)
  – Takashi Furumura (ERI/U.Tokyo)
  – Hiroshi Okuda (GSFS/U.Tokyo)
  – Takeshi Iwashita (Kyoto U., ITC/Hokkaido U.)
  – Hide Sakaguchi (IFREE/JAMSTEC)

• **Main Members**
  – Takahiro Katagiri (ITC/U.Tokyo)
  – Masaharu Matsumoto (ITC/U.Tokyo)
  – Hideyuki Jitsumoto (Tokyo Tech)
  – Satoshi Ohshima (ITC/U.Tokyo)
  – Hiroyasu Hasumi (AORI/U.Tokyo)
  – Takashi Arakawa (RIST)
  – Futoshi Mori (ERI/U.Tokyo)
  – Takeshi Kitayama (GSFS/U.Tokyo)
  – Akihiro Ida (ACCMS/Kyoto U.)
  – Miki Yamamoto (IFREE/JAMSTEC)
  – Daisuke Nishiura (IFREE/JAMSTEC)
User’s Program

Framework
- ppOpen-APPL
- FEM
- FDM
- FVM
- BEM
- DEM

Appl. Dev.
- ppOpen-MATH
- MG
- GRAPH
- VIS
- MP

Math
Libraries
- ppOpen-AT
- STATIC
- DYNAMIC

Automatic
Tuning (AT)
- ppOpen-SYS
- COMM
- FT

System
Software

ppOpen-HPC

Optimized Application with
Optimized ppOpen-APPL, ppOpen-MATH
ppOpen-HPC covers …

FEM
Finite Element Method

FDM
Finite Difference Method

FVM
Finite Volume Method

BEM
Boundary Element Method

DEM
Discrete Element Method
ppOpen-HPC: ppOpen-APPL

- ppOpen-HPC consists of various types of *optimized* libraries, which covers various types of procedures for scientific computations.
  - ppOpen-APPL/FEM, FDM, FVM, BEM, DEM
  - Linear Solvers, Mat. Assemble, AMR, Visualization etc.
  - written in Fortran 2003 (C interface is available soon)
- Source code developed on a PC with a single processor is linked with these libraries, and generated parallel code is optimized for post-peta scale system.
  - Based on components extracted from existing REAL applications on K and large-scale computers
  - Users don’t have to worry about optimization tuning, parallelization etc.
    - Part of MPI, OpenMP, (OpenACC)
Supercomputers in U.Tokyo

2 big systems, 6 yr. cycle

FY
05 06 07 08 09 10 11 12 13 14 15 16 17 18 19

Hitachi SR11000/J2
18.8 TFLOPS, 16.4 TB
Fat nodes with large memory

Hitachi SR16000/M1 based on IBM Power-7
54.9 TFLOPS, 11.2 TB
Our last SMP, to be switched to MPP

Hitachi HA8000 (T2K)
140 TFLOPS, 31.3 TB
(Flat) MPI, good comm. performance

Fujitsu PRIMEHPC FX10 based on SPARC64 IXfx
1.13 PFLOPS, 150 TB
Turning point to Hybrid Parallel Prog. Model

Post T2K
20-30 PFLOPS

Initial Plan

Peta

京(=K)
Target of ppOpen-HPC: Post T2K System

• Target system is Post T2K system
  – > 30 PFLOPS, Fall 2016
    ✓ JCAHPC (Joint Center for Advanced High Performance Computing): U. Tsukuba & U. Tokyo
    ✓ http://jcahpc.jp/
  – Many-core based (e.g. Intel MIC/Xeon Phi)
    ✓ MPI + OpenMP + X
  – ppOpen-HPC helps smooth transition of users (> 2,000) to new system

• K/FX10, Cray, Xeon clusters are also in scope
Schedule of Public Release
(with English Documents, MIT License)
http://ppopenhpc.cc.u-tokyo.ac.jp/

• Released at SC-XY (or can be downloaded)
• Multicore/manycore cluster version (Flat MPI, OpenMP/MPI Hybrid) with documents in English
• We are now focusing on MIC/Xeon Phi
• Collaborations are welcome

• History
  – SC12, Nov 2012 (Ver.0.1.0)
  – SC13, Nov 2013 (Ver.0.2.0)
  – SC14, Nov 2014 (Ver.0.3.0)
  – SC15, Nov 2015 (Ver.1.0.0) (plan)
New Features in Ver.0.3.0
http://ppopenhpc.cc.u-tokyo.ac.jp/

- ppOpen-APPL/AMR-FDM: AMR framework with a dynamic load-balancing method for various FDM applications
- HACApk library for H-matrix computations in ppOpen-APPL/BEM
- Utilities for pre-processing in ppOpen-APPL/DEM
Collaborations, Outreaching

• Collaborations
  – International Collaborations
    • Lawrence Berkeley National Lab.
    • National Taiwan University
    • ESSEX/SPPEXA/DFG, Germany
    • IPCC (Intel Parallel Computing Center)

• Outreaching, Applications
  – Large-Scale Simulations
    • Geologic CO$_2$ Storage
    • Astrophysics
    • Earthquake Simulations etc.
    • ppOpen-AT, ppOpen-MATH/VIS, ppOpen-MATH/MP, Linear Solvers
  – Intl. Workshops (2012, 13, 15)
  – Tutorials, Classes
• ppOpen-HPC
• ppOpen-MATH
  – ppOpen-MATH/MP: Coupler
• ppOpen-AT
User’s Program

ppOpen-APPL  FEM  FDM  FVM  BEM  DEM

ppOpen-MATH  MG  GRAPH  VIS  MP

ppOpen-AT  STATIC  DYNAMIC

ppOpen-SYS  COMM  FT

ppOpen-HPC

Optimized Application with Optimized ppOpen-APPL, ppOpen-MATH
ppOpen-MATH

• A set of common numerical libraries
  – Multigrid solvers (ppOpen-MATH/MG)
  – Parallel graph libraries (ppOpen-MATH/GRAPH)
    • Multithreaded RCM for reordering (under development)
  – Parallel visualization (ppOpen-MATH/VIS)
  – Library for coupled multi-physics simulations (loose-coupling) (ppOpen-MATH/MP)
    • Originally developed as a coupler for NICAM (atmosphere, unstructured), and COCO (ocean, structured) in global climate simulations using K computer
      – Both codes are major codes on the K computer.
        » Prof. Masaki Satoh (AORI/U.Tokyo): NICAM
        » Prof. Hiroyasu Hasumi (AORI/U.Tokyo): COCO
    • Developed coupler is extended to more general use.
      – Coupled seismic simulations
NICAM: Semi-Unstructured Grid
MIROC-A: FDM/Structured Grid

NICAM-Agrid
NICAM-ZMgrid

Atmospheric Model-1
Atmospheric Model-2

ppOpen-MATH/MP Coupler

• Grid Transformation
• Multi-Ensemble
• IO
• Pre- and post-process
• Fault tolerance
• M×N

Ocean Model

COCO Regional COCO Matsumura-model

COCO: Tri-Polar FDM
Regional Ocean Model Non Hydrostatic Model

J-cup

Post-Peta-Scale System
- System S/W
- Architecture

c/o T. Arakawa, M. Satoh
Sea surface temperature (OSST)

left: COCO (Ocean: Structured), right: NICAM (Atmospheric: Semi-Unst.)
Thickness of Sea Ice (OHI)

left: COCO (Ocean: Structured), right: NICAM (Atmospheric: Semi-Unst.)
Weak-Coupled Simulation by the ppOpen-HPC Libraries

Two kinds of applications (Seism3D+ based on FDM, and FrontISTR++ based on FEM) are connected by the ppOpen-MATH/MP coupler.

Seism3D+

ppOpen-APPL/FDM

Velocity

ppOpen-MATH/MP

ppOpen-APPL/FEM

FrontISTR++

Displacement

Principal Functions

- Make a mapping table
- Convert physical variables
- Choose a timing of data transmission
Two Features of ppOpen-MATH/MP

1. Execution pattern is NOT a standalone process

To save computational resources, the ppOpen-MATH/MP is executed as not an application but a library.

2. Applicable to various discretization methods*

The ppOpen-MATH/MP has wide applicability because it is independent from grid structure and enables users to implement an arbitrary interpolation code.

*Users must prepare the interpolation code and a correspondence table as input information.
Dataflow of ppOpen-MATH/MP*

1. Data-packing into a buffer

2. Send-data extraction from the buffer, and data sending

3. Data-packing after the interpolation process

4. Data extraction from the buffer

* Also applicable to full coupling, multiple applications
The coupling simulation refers to one-way data communication from FDM (seismic wave propagation) to FEM (dynamic structure).
Numerical Model Description

**Seism3D+** (composed of ppOpen-APPL/FDM)

Explicit FDM Application for Seismic Wave Analysis

\[
\rho \frac{\partial v_p}{\partial t} = \left( \frac{\partial \sigma_{xp}}{\partial x} + \frac{\partial \sigma_{yp}}{\partial y} + \frac{\partial \sigma_{zp}}{\partial z} + f_p \right), \quad (p = x, y, z)
\]

\[
\frac{\partial \sigma_{pq}}{\partial t} = \lambda \left( \frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} + \frac{\partial v_z}{\partial z} \right) \delta_{pq} + \mu \left( \frac{\partial v_p}{\partial q} + \frac{\partial v_q}{\partial p} \right), \quad (p, q = x, y, z)
\]

**FrontISTR++** (composed of ppOpen-APPL/FEM)

Implicit FEM Application for Structural Analysis

\[
M\ddot{d} + C\dot{d} + Kd = F
\]

- **M**: mass matrix
- **C**: damping matrix
- **K**: stiffness matrix
- **F**: nodal load vector
- **d**: nodal displacement vector

Computational load: **Seism3D+ < FrontISTR++**
Implementation of the Coupling Simulation

Unstructured mesh used in FrontISTR++
Colors: MPI process (64 processes)
(Partitioning by METIS)
Practical Simulation on Oakleaf-FX

The simulation target is the earthquake that occurred at Awaji Island on 13 April, 2013. The computational domain of Seism3D+ is 60 km² from Awaji Island and that of FrontISTR++ is the actual building of RIKEN Advanced Institute for Computational Science (AICS), Port Island, Kobe, modeled by an unstructured mesh.

**Seism3D+**
- Grid Points: \((x, y, z) = (1536, 1536, 1600)\)
- Parallelization: 2560 processes/16 threads

**FrontISTR++**
- Grid Points: 600 million (AICS building)
- Parallelization:
  - 1000 processes/16 threads (@Port Island)
  - 1000 processes/16 threads (@Kobe Stadium)

Total 4560 nodes on Oakleaf-FX
(Seism3D+: 2560 nodes, FrontISTR++: 2000 nodes)
2,560 nodes for FDM, 2,000 nodes for FEM = 4,560 nodes of FX10

Seismic wave propagation by Seism3D+ (Red: P-wave, Green: S-wave)

Building vibration by FrontISTR++
2,560 nodes for FDM, 2,000 nodes for FEM = 4,560 nodes of FX10

- Coupling simulation was executed on large-scale computational resources of Oakleaf-FX supercomputer system.

- Seismic wave propagations (Seism3D+) for the simulation time of 90 sec., and building vibrations (FrontISTR++) for the simulation time of 20 sec. were calculated.

<table>
<thead>
<tr>
<th></th>
<th>Sim. Time</th>
<th>Exe. Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seism3D+</td>
<td>90 sec.</td>
<td>6 hours</td>
</tr>
<tr>
<td>FrontISTR++</td>
<td>20 sec.</td>
<td>16 hours</td>
</tr>
</tbody>
</table>

- It was revealed that the manner in which memory allocation occurs in the coupler has some problem when such a large-scale simulation is performed.
The implementation of a multi-scale coupling simulation of seismic waves and building vibrations was introduced and a performance evaluation was conducted. Finally, a practical simulation on the Oakleaf-FX was performed.

Although more research needs to be conducted, such a multi-scale coupling simulation will be important for numerical simulations on next generation of large-scale computational environments.
• ppOpen-HPC
• ppOpen-MATH
  – ppOpen-MATH/MP: Coupler
• ppOpen-AT
ppOpen-AT

• Automatic Tuning (AT) enables development of optimized codes and libraries on emerging architectures

• ppOpen-AT automatically and adaptively generates optimum implementation for efficient memory accesses in procedures of scientific computing in each component of ppOpen-APPL

• A directive-based special AT language is developed
  – Well-known loop transformation techniques are utilized

• ppOpen-APPL/FDM, ppOpen-APPL/BEM

• AT applied to 3D FDM code for seismic simulations developed on ppOpen-APPL/FDM for Intel Xeon/Phi
ppOpen-AT System

1. User Knowledge
2. ppOpen-AT Directives
3. ppOpen-APPL /*
4. Selection
5. ppOpen-AT Auto-Tuner
6. Auto-tuned Kernel Execution

Before Release-time
Automatic Code Generation

Library Developer
Library User

Library Call

Execution Time
: Target Computers

c/o T.Katagiri

Candidate n
Candidate 3
Candidate 2
Candidate 1
Seism3D on ppOpen-APPL/FDM

- **Space difference by FDM.**
  \[
  \frac{d}{dx} \sigma_{pq}(x, y, z) = \frac{1}{\Delta x} \sum_{m=1}^{M/2} c_m [\sigma_{pq} \{x + (m + \frac{1}{2})\Delta x, y, z\} - \sigma_{pq} \{x - (m - \frac{1}{2})\Delta x, y, z\}], \\
  (p, q = x, y, z)
  \]

- **Explicit time expansion by central difference.**
  \[
  u_p^{n+\frac{1}{2}} = u_p^{n-\frac{1}{2}} + \frac{1}{\rho} \left( \frac{\partial \sigma_{sp}^n}{\partial x} + \frac{\partial \sigma_{sp}^n}{\partial y} + \frac{\partial \sigma_{sp}^n}{\partial z} + f_p^n \right) \Delta t, (p = x, y, z)
  \]
Seism3D: Code for Seismic Wave Sim.
Triple-nested loops, Most Expensive

```plaintext
DO K = 1, NZ
  DO J = 1, NY
    DO I = 1, NX
      RL  = LAM(I,J,K)
      RM  = RIG(I,J,K)
      RM2 = RM + RM
      RMAXY = 4.0/(1.0/RIG(I,J,K) + 1.0/RIG(I+1,J,K) + 1.0/RIG(I,J+1,K) + 1.0/RIG(I+1,J+1,K))
      RMAXZ = 4.0/(1.0/RIG(I,J,K) + 1.0/RIG(I+1,J,K) + 1.0/RIG(I,J+1,K) + 1.0/RIG(I+1,J+1,K))
      RMAYZ = 4.0/(1.0/RIG(I,J,K) + 1.0/RIG(I+1,J,K) + 1.0/RIG(I,J+1,K) + 1.0/RIG(I+1,J+1,K))
      RLTHETA = (DXVX(I,J,K)+DYVVY(I,J,K)+DZVZ(I,J,K))*RL
      QG   = ABSX(I)*ABSY(J)*ABSZ(K)*Q(I,J,K)
      SXX (I,J,K) = ( SXX (I,J,K) + (RLTHETA + RM2*DXVX(I,J,K))*DT )*QG
      SYY (I,J,K) = ( SYY (I,J,K) + (RLTHETA + RM2*DYVY(I,J,K))*DT )*QG
      SZZ (I,J,K) = ( SZZ (I,J,K) + (RLTHETA + RM2*DZVZ(I,J,K))*DT )*QG
      SXY (I,J,K) = ( SXY (I,J,K) + (RMAXY*(DXVY(I,J,K)+DYVX(I,J,K)))*DT )*QG
      SXZ (I,J,K) = ( SXZ (I,J,K) + (RMAXZ*(DXVZ(I,J,K)+DZVX(I,J,K)))*DT )*QG
      SYZ (I,J,K) = ( SYZ (I,J,K) + (RMAYZ*(DYVZ(I,J,K)+DZVY(I,J,K)))*DT )*QG
      END DO
    END DO
  END DO
END DO
```

c/o T.Katagiri
DO K = 1, NZ
DO J = 1, NY
DO I = 1, NX
  RL  = LAM (I,J,K)
  RM  = RIG (I,J,K)
  RM2 = RM + RM
  RLTTHETA = (DXVX(I,J,K)+DYVY(I,J,K)+DZVZ(I,J,K))*RL
  QG  = ABSX(I)*ABSY(J)*ABSZ(K)*Q(I,J,K)
  SXX (I,J,K) = ( SXX (I,J,K) + (RLTTHETA + RM2*DXVX(I,J,K))*DT )*QG
  SYY (I,J,K) = ( SYY (I,J,K) + (RLTTHETA + RM2*DYVY(I,J,K))*DT )*QG
  SZZ (I,J,K) = ( SZZ (I,J,K) + (RLTTHETA + RM2*DZVZ(I,J,K))*DT )*QG
END DO; END DO; END DO;

DO K = 1, NZ
DO J = 1, NY
DO I = 1, NX
  STMP1 = 1.0/RIG(I,J,K)
  STMP2 = 1.0/RIG(I+1,J,K)
  STMP3 = STMP1 + STMP2
  RMAXY = 4.0/(STMP3 + 1.0/RIG(I,J+1,K) + 1.0/RIG(I+1,J+1,K))
  RMAXZ = 4.0/(STMP3 + STMP4 + 1.0/RIG(I+1,J,K+1))
  RMAYZ = 4.0/(STMP3 + STMP4 + 1.0/RIG(I+1,J,K+1))
  QG  = ABSX(I)*ABSY(J)*ABSZ(K)*Q(I,J,K)
  SXY (I,J,K) = ( SXY (I,J,K) + (RMAXY*(DXVY(I,J,K)+DYVX(I,J,K)))*DT )*QG
  SXZ (I,J,K) = ( SXZ (I,J,K) + (RMAXZ*(DXVZ(I,J,K)+DZVX(I,J,K)))*DT )*QG
  SYZ (I,J,K) = ( SYZ (I,J,K) + (RMAYZ*(DYVZ(I,J,K)+DZVY(I,J,K)))*DT )*QG
END DO; END DO; END DO;

/* c/o T.Katagiri */
DO KK = 1, NZ * NY  
  K = (KK-1)/NY + 1  
  J = mod(KK-1,NY) + 1  
DO I = 1, NX  
  RL  = LAM(I,J,K)  
  RM  = RIG(I,J,K)  
  RM2 = RM + RM  
  RMAXY = 4.0/(1.0/RIG(I,J,K) + 1.0/RIG(I+1,J,K) + 1.0/RIG(I,J+1,K) + 1.0/RIG(I+1,J+1,K))  
  RMAXZ = 4.0/(1.0/RIG(I,J,K) + 1.0/RIG(I+1,J,K) + 1.0/RIG(I,J,K+1) + 1.0/RIG(I+1,J,K+1))  
  RMAYZ = 4.0/(1.0/RIG(I,J,K) + 1.0/RIG(I+1,J,K) + 1.0/RIG(I,J,K+1) + 1.0/RIG(I+1,J,K+1))  
  RLTHETA = (DXVX(I,J,K)+DYVY(I,J,K)+DZVZ(I,J,K))*RL  
  QG  = ABSX(I)*ABSY(J)*ABSZ(K)*Q(I,J,K)  
  SXX (I,J,K) = ( SXX (I,J,K) + (RLTHETA + RM2*DXVX(I,J,K))*DT )*QG  
  SYY (I,J,K) = ( SYY (I,J,K) + (RLTHETA + RM2*DYVY(I,J,K))*DT )*QG  
  SZZ (I,J,K) = ( SZZ (I,J,K) + (RLTHETA + RM2*DZVZ(I,J,K))*DT )*QG  
  SXY (I,J,K) = ( SXY (I,J,K) + (RMAXX*(DXVY(I,J,K)+DYVX(I,J,K)))*DT )*QG  
  SXZ (I,J,K) = ( SXZ (I,J,K) + (RMAXZ*(DXVZ(I,J,K)+DZVX(I,J,K)))*DT )*QG  
  SYZ (I,J,K) = ( SYZ (I,J,K) + (RMAYZ*(DYVZ(I,J,K)+DZVY(I,J,K)))*DT )*QG  
ENDDO  
END DO  

Loop Fusion: Double-nested  

Longer loops  

Inner loop: nice for prefetching  

c/o T.Katagiri
Example of Directives of ppOpen-AT

!oat$ install LoopFusionSplit region start
$omp parallel do
private(k,j,i,STMP1,STMP2,STMP3,STMP4,RL,RM,RM2,RMAXY,RMAXZ,RMAYZ,RLTHETA,QG)
  DO K = 1, NZ
    DO J = 1, NY
      DO I = 1, NX
        RL  = LAM (I,J,K);   RM  = RIG (I,J,K);   RM2 = RM + RM
        RLTHETA  = (DXVX(I,J,K)+DYVY(I,J,K)+DZVZ(I,J,K))*RL
        QG  = ABSX(I)*ABSY(J)*ABSZ(K)*Q(I,J,K)
      !oat$ SplitPointCopyDef region start
      SXX (I,J,K) = ( SXX (I,J,K) + (RLTHETA + RM2*DXVX(I,J,K))*DT )*QG
      SYY (I,J,K) = ( SYY (I,J,K) + (RLTHETA + RM2*DYVY(I,J,K))*DT )*QG
      SZZ (I,J,K) = ( SZZ (I,J,K) + (RLTHETA + RM2*DZVZ(I,J,K))*DT )*QG
    !oat$ SplitPoint (K, J, I)
      STMP1 = 1.0/RIG(I,J,K);  STMP2 = 1.0/RIG(I+1,J,K);  STMP4 = 1.0/RIG(I,J,K+1)
      STMP3 = STMP1 + STMP2
      RMAXY = 4.0/(STMP3 + 1.0/RIG(I,J+1,K) + 1.0/RIG(I+1,J+1,K))
      RMAXZ = 4.0/(STMP3 + STMP4 + 1.0/RIG(I,J+1,K+1))
      RMAYZ = 4.0/(STMP3 + STMP4 + 1.0/RIG(I+1,J,K+1))
      !oat$ SplitPointCopyInsert
      SXY (I,J,K) = ( SXY (I,J,K) + (RMAXY*(DXVY(I,J,K)+DYVX(I,J,K)))DT )*QG
      SXZ (I,J,K) = ( SXZ (I,J,K) + (RMAXZ*(DXVZ(I,J,K)+DZVX(I,J,K)))DT )*QG
      SYZ (I,J,K) = ( SYZ (I,J,K) + (RMAYZ*(DYVZ(I,J,K)+DZVY(I,J,K)))DT )*QG
      END DO; END DO; END DO
  !oat$ SplitPointCopyDef region end
$omp end parallel do
!oat$ install LoopFusionSplit region end
Automatic Generated Codes for the kernel 1

ppohFDM_update_stress

- #1 [Baseline]: Original 3-nested Loop
- #2 [Split]: Loop Splitting with K-loop (Separated, two 3-nested loops)
- #3 [Split]: Loop Splitting with J-loop
- #4 [Split]: Loop Splitting with I-loop
- #5 [Split&Fusion]: Loop Fusion to #1 for K and J-loops (2-nested loop)
- #6 [Split&Fusion]: Loop Fusion to #2 for K and J-Loops (2-nested loop)
- #7 [Fusion]: Loop Fusion to #1 (loop collapse)
- #8 [Split&Fusion]: Loop Fusion to #2 (loop collapse, two one-nest loop)
Automatic Generated Codes for the kernel 2

ppohFDM_update_vel

• #1 [Baseline]: Original 3-nested Loop.
• #2 [Fusion]: Loop Fusion for K and J-Loops. (2-nested loop)
• #3 [Fusion]: Loop Split for K, J, and I-Loops. (Loop Collapse)
• #4 [Fusion&Re-order]: Re-ordering of sentences to #1.
• #5 [Fusion&Re-order]: Re-ordering of sentences to #2.
• #6 [Fusion&Re-order]: Re-ordering of sentences to #3.
ppOpen-AT/Static, without AT on 8-nodes of Xeon Phi (1,920 threads)

P: # MPI processes
T: # Threads/Process

Comp. Time vs. Comm. Time

P8T240 P16T120 P32T60 P64T30 P128T15 P240T8 P480T4
ppOpen-AT/Static, with AT

12.1% Speedups

P: # MPI processes
T: # Threads/Process

Comp. Time
Comm. Time

P8T240 P16T120 P32T60 P64T30 P128T15 P240T8 P480T4
Example of directive for ppOpen-AT
Loop splitting/fusion

Effect of AT on each kernel
(Xeon Phi 8-nodes)
## Target H/W

<table>
<thead>
<tr>
<th>Code Name</th>
<th>FX10</th>
<th>MIC</th>
<th>IvyB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architectures</td>
<td>Fujitsu SPARC64 IX fx</td>
<td>Intel Xeon Phi 5110P (Knights Corner)</td>
<td>Intel Xeon E5-2680 v2 (Ivy-Bridge-EP)</td>
</tr>
<tr>
<td>Frequency (GHz)</td>
<td>1.848</td>
<td>1.053</td>
<td>2.80</td>
</tr>
<tr>
<td>Core # (Thread #)</td>
<td>16 (16)</td>
<td>60 (240)</td>
<td>10 (20)</td>
</tr>
<tr>
<td>Memory Type</td>
<td>DDR3</td>
<td>GDDR5</td>
<td>DDR3</td>
</tr>
<tr>
<td>Peak Performance (GFLOPS)</td>
<td>236.5</td>
<td>1,010.9</td>
<td>224.0</td>
</tr>
<tr>
<td>without AT (Flat MPI) (GFLOPS/socket)</td>
<td>20.0</td>
<td>19.4</td>
<td>20.4</td>
</tr>
</tbody>
</table>
**Speed-up by AT**

<table>
<thead>
<tr>
<th>Code Name</th>
<th>FX10</th>
<th>MIC</th>
<th>IvyB</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Architectures</strong></td>
<td>Fujitsu SPARC64 IX fx</td>
<td>Intel Xeon Phi 5110P (Knights Corner)</td>
<td>Intel Xeon E5-2680 v2 (Ivy-Bridge-EP)</td>
</tr>
<tr>
<td>Frequency (GHz)</td>
<td>1.848</td>
<td>1.053</td>
<td>2.80</td>
</tr>
<tr>
<td>Core # (Thread #)</td>
<td>16 (16)</td>
<td>60 (240)</td>
<td>10 (20)</td>
</tr>
<tr>
<td>Memory Type</td>
<td>DDR3</td>
<td>GDDR5</td>
<td>DDR3</td>
</tr>
<tr>
<td>Peak Performance (GFLOPS)</td>
<td>236.5</td>
<td>1,010.9</td>
<td>224.0</td>
</tr>
<tr>
<td><strong>without AT (Flat MPI) (GFLOPS/socket)</strong></td>
<td>20.0</td>
<td>19.4</td>
<td>20.4</td>
</tr>
<tr>
<td><strong>with AT (GFLOPS/socket)</strong></td>
<td><strong>20.3</strong> P128T1</td>
<td><strong>30.0</strong> P240T8</td>
<td><strong>23.4</strong> P80T1</td>
</tr>
</tbody>
</table>
Features & Future Works in ppOpen-AT

• Strongly depends on intelligence/experiences of the users: manual operations ... but it is ok (education)
  – Configurations of scenarios, Data dependency
  – Locations of directives
  – Effects of problem size, hardware parameters etc.
  – Parallel prog. model (#MPI processes x #OpenMP threads)

• Analyses of assembly codes (for research paper)

• Operations for sparse matrices (not limited to SpMV)
  – Blocking + X-**ELL**-Y-Z (e.g. **SELL-C-**σ)
  – Different from FDM kernels

• **Matrix Assembly of FEM**

• DSL’s for stencil computing: Physis, ExaStencil etc.