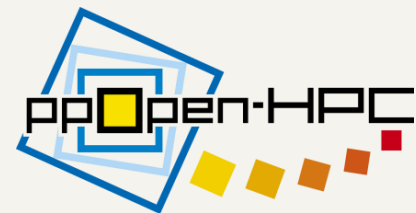


第52回お試しアカウント付き並列プログラミング講習会
「ライブラリ利用: 高性能プログラミング初級入門」

2016.02.03-02.04@東京大学情報基盤センター



差分法による弾性波動並列シミュレーション

岩手医科大学 医歯薬総合研究所

森 太志
(fumori@iwate-med.ac.jp)

東京大学 地震研究所

古村 孝志

- はじめに
 - 近年のスパコンのトレンド
 - ppOpen-APPL/FDM libraryの概要
 - 弾性波動計算
 - ヘテロな環境におけるパフォーマンステスト
 - 応用例: 大規模連成計算
- 演習: ppOpen-APPL/FDM
 - 利用方法
 - 課題

TOP500から見る スーパーコンピュータの傾向



RANK	SITE	SYSTEM	CORES	RMAX (TFLOP/S)	RPEAK (TFLOP/S)	POWER (KW)
1	National Super Computer Center in Guangzhou China	Tianhe-2 (MilkyWay-2) - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, <u>Intel Xeon Phi 31S1P</u> , NUDT	3,120,000	33,862.7	54,902.4	17,808
2	DOE/SC/Oak Ridge National Laboratory United States	Titan - Cray XK7, Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, <u>NVIDIA K20x</u> , Cray Inc.	560,640	17,590.0	27,112.5	8,209
3	DOE/NNSA/LLNL United States	Sequoia - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom IBM	1,572,864	17,173.2	20,132.7	7,890
4	RIKEN Advanced Institute for Computational Science (AICS) Japan	K computer , SPARC64 VIIIfx 2.0GHz, Tofu interconnect Fujitsu	705,024	10,510.0	11,280.4	12,660
5	DOE/SC/Argonne National Laboratory United States	Mira - BlueGene/Q, Power BQC 16C 1.60GHz, Custom IBM	786,432	8,586.6	10,066.3	3,945
6	DOE/NNSA/LANL/SNL United States	Trinity - Cray XC40, <u>Xeon E5-2698v3</u> 16C 2.3GHz, Aries interconnect Cray Inc.	301,056	8,100.9	11,078.9	
7	Swiss National Supercomputing Centre (CSCS) Switzerland	Piz Daint - Cray XC30, Xeon E5-2670 8C 2.600GHz, Aries interconnect, <u>NVIDIA K20x</u> , Cray Inc.	115,984	6,271.0	7,788.9	2,325
8	HLRS - Höchstleistungsrechenzentrum Stuttgart Germany	Hazel Hen - Cray XC40, <u>Xeon E5-2680v3</u> 12C 2.5GHz, Aries interconnect Cray Inc.	185,088	5,640.2	7,403.5	
9	King Abdullah University of Science and Technology Saudi Arabia	Shaheen II - Cray XC40, <u>Xeon E5-2698v3</u> 16C 2.3GHz, Aries interconnect Cray Inc.	196,608	5,537.0	7,235.2	2,834
10	Texas Advanced Computing Center/Univ. of Texas United States	Stampede - PowerEdge C8220, Xeon E5-2680 8C 2.700GHz, Infiniband FDR, <u>Intel Xeon Phi SE10P</u> , Dell	462,462	5,168.1	8,520.1	4,510

From Top 500 (2015.11)

Many Coreや

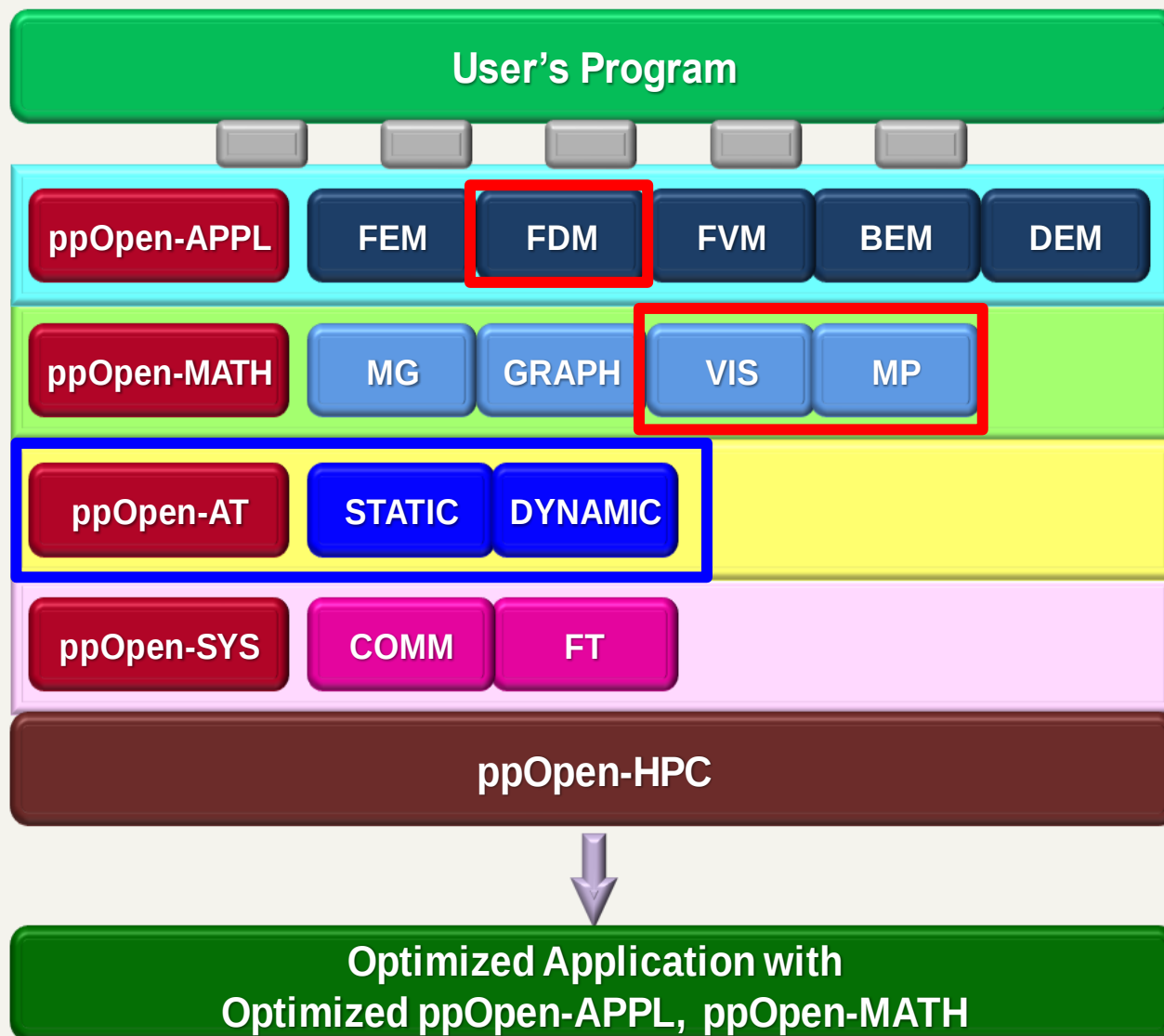
Many Core + Accelerator / Co-processor を搭載しているシステムの変動

特にMany Core + Accelerator / Co-processorは、75システム(2014.11)から104システム(2015.11)に増加。内4システムが両方搭載



<http://www.intel.co.jp/>

Post T2KはIntel Xeon Phi Co-processorを搭載予定



ppOpen-APPL/FDM library

概要

① 運動方程式

$$\rho \ddot{u} = \frac{\partial \sigma_{xp}}{\partial x} + \frac{\partial \sigma_{yp}}{\partial y} + \frac{\partial \sigma_{zp}}{\partial z} + f_p, (p = x, y, z) \quad (1)$$

$\ddot{u}$: 加速度、 σ : 応力、 ρ : 密度、 f : 外力

② 弾性体の無限小変位によるひずみテンソルは以下の式で示せる

$$e_{pq} = \frac{1}{2} \left(\frac{\partial u_p}{\partial q} + \frac{\partial u_q}{\partial p} \right), (p, q = x, y, z) \quad (2)$$

e : 歪み

等方弾性体の場合には応力テンソルは以下の式で示せる

$$\sigma_{pq} = \lambda(e_{xx} + e_{yy} + e_{zz})\delta_{pq} + 2\mu e_{pq}, (p, q = x, y, z) \quad (3)$$

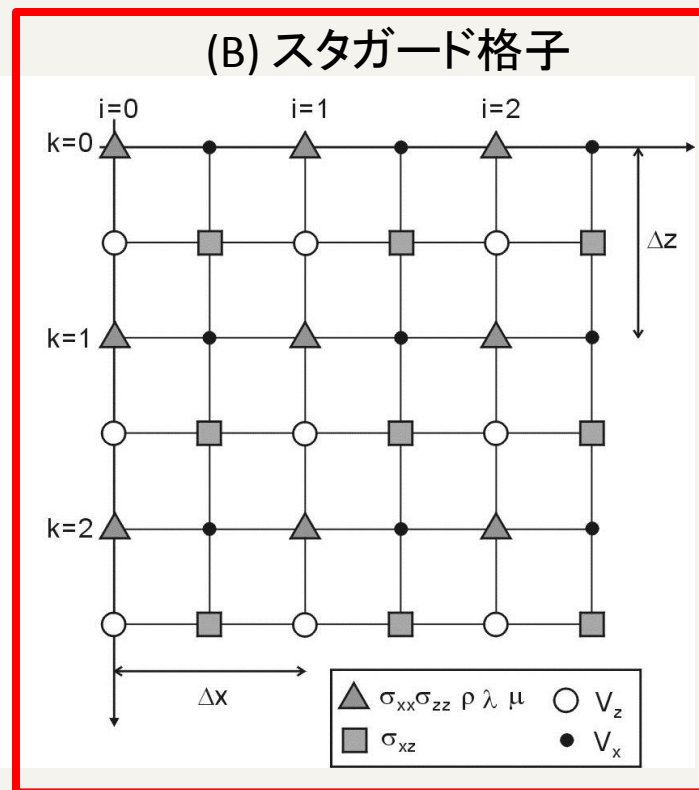
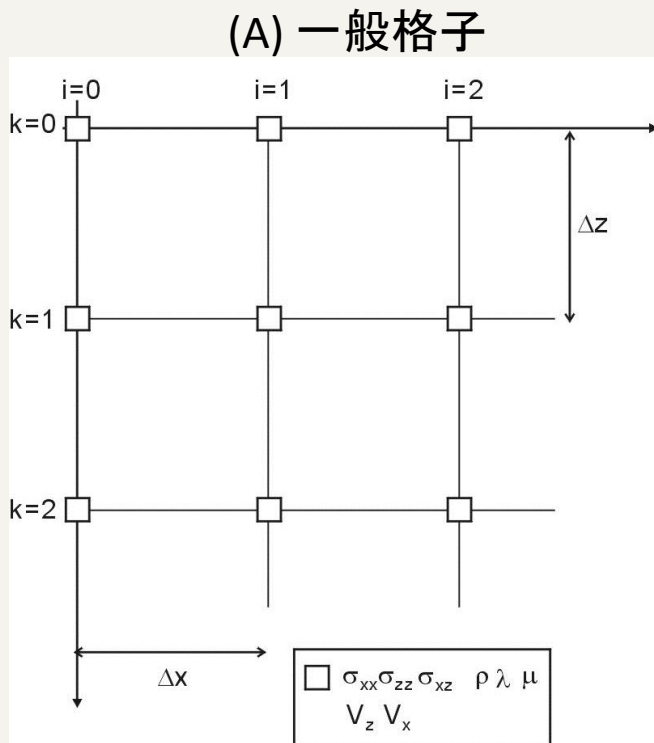
λ, μ : Lamé定数、 δ : クロネッカのデルタ

運動方程式と構成方程式

$$\left. \begin{aligned} \rho \frac{\partial v_x}{\partial t} &= \frac{\partial \sigma_{xx}}{\partial x} + \frac{\partial \sigma_{yx}}{\partial y} + \frac{\partial \sigma_{zx}}{\partial z} + f_x \\ \rho \frac{\partial v_y}{\partial t} &= \frac{\partial \sigma_{xy}}{\partial x} + \frac{\partial \sigma_{yy}}{\partial y} + \frac{\partial \sigma_{zy}}{\partial z} + f_y \\ \rho \frac{\partial v_z}{\partial t} &= \frac{\partial \sigma_{xz}}{\partial x} + \frac{\partial \sigma_{yz}}{\partial y} + \frac{\partial \sigma_{zz}}{\partial z} + f_z \end{aligned} \right\} \text{運動方程式} \quad (4)$$

$$\begin{aligned} \frac{\partial \sigma_{xx}}{\partial t} &= (\lambda + 2\mu) \frac{\partial v_x}{\partial x} + \lambda \left(\frac{\partial v_y}{\partial y} + \frac{\partial v_z}{\partial z} \right) + g_{xx} & \frac{\partial \sigma_{yz}}{\partial t} &= \mu \left(\frac{\partial v_y}{\partial z} + \frac{\partial v_z}{\partial y} \right) + g_{yz} \\ \frac{\partial \sigma_{yy}}{\partial t} &= (\lambda + 2\mu) \frac{\partial v_y}{\partial y} + \lambda \left(\frac{\partial v_x}{\partial x} + \frac{\partial v_z}{\partial z} \right) + g_{yy} & \frac{\partial \sigma_{xz}}{\partial t} &= \mu \left(\frac{\partial v_x}{\partial z} + \frac{\partial v_z}{\partial x} \right) + g_{xz} \\ \frac{\partial \sigma_{zz}}{\partial t} &= (\lambda + 2\mu) \frac{\partial v_z}{\partial z} + \lambda \left(\frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} \right) + g_{zz} & \frac{\partial \sigma_{xy}}{\partial t} &= \mu \left(\frac{\partial v_x}{\partial y} + \frac{\partial v_y}{\partial x} \right) + g_{xy} \end{aligned} \quad (5)$$

構成方程式



- A) 変位や応力、物性値など全ての変数を同一格子点上に配置する一般格子
- B) 変位を半格子ずれた位置に定義するスタガード格子
 - 変数の微分が定義される位置に関連の変数が位置するため計算精度が良い

• 空間微分

– 中心差分による計算(2次精度、4次精度、8次精度)

$$(2\text{次精度}) \quad \frac{d}{dx} \sigma_{pq}(x, y, z) \simeq \frac{1}{\Delta x} \left[\sigma_{pq} \left(x + \frac{\Delta x}{2}, y, z \right) - \sigma_{pq} \left(x - \frac{\Delta x}{2}, y, z \right) \right]$$

$$(4\text{次精度}) \quad \frac{d}{dx} \sigma_{pq}(x, y, z) \simeq \frac{1}{\Delta x} \left[\frac{9}{8} \left\{ \sigma_{pq} \left(x + \frac{\Delta x}{2}, y, z \right) - \sigma_{pq} \left(x - \frac{\Delta x}{2}, y, z \right) \right\} \right. \\ \left. - \frac{1}{24} \left\{ \sigma_{pq} \left(x + \frac{3\Delta x}{2}, y, z \right) - \sigma_{pq} \left(x - \frac{3\Delta x}{2}, y, z \right) \right\} \right]$$

$$(8\text{次精度}) \quad \frac{d}{dx} \sigma_{pq}(x, y, z) \simeq \frac{1}{\Delta x} \left[\frac{1225}{1024} \left\{ \sigma_{pq} \left(x + \frac{\Delta x}{2}, y, z \right) - \sigma_{pq} \left(x - \frac{\Delta x}{2}, y, z \right) \right\} \right. \\ - \frac{245}{3072} \left\{ \sigma_{pq} \left(x + \frac{3\Delta x}{2}, y, z \right) - \sigma_{pq} \left(x - \frac{3\Delta x}{2}, y, z \right) \right\} \\ + \frac{49}{5120} \left\{ \sigma_{pq} \left(x + \frac{5\Delta x}{2}, y, z \right) - \sigma_{pq} \left(x - \frac{5\Delta x}{2}, y, z \right) \right\} \\ \left. - \frac{5}{7168} \left\{ \sigma_{pq} \left(x + \frac{7\Delta x}{2}, y, z \right) - \sigma_{pq} \left(x - \frac{7\Delta x}{2}, y, z \right) \right\} \right]$$

- 時間を $t = t_0 + n\Delta t$ ($n = 1, 2, \dots$) と離散化し、中心差分により時間積分を行う。
- 速度と応力それぞれについて互いに半グリッドずれたスタガード格子を採用し、以下の式のように評価する。

$$\dot{u}_p^{n+\frac{1}{2}} = \dot{u}_p^{n-\frac{1}{2}} + \frac{1}{\rho} \left(\frac{\partial \sigma_{xp}^n}{\partial x} + \frac{\partial \sigma_{yp}^n}{\partial y} + \frac{\partial \sigma_{zp}^n}{\partial z} + f_p^n \right) \Delta t, \quad (p = x, y, z) \quad (6)$$

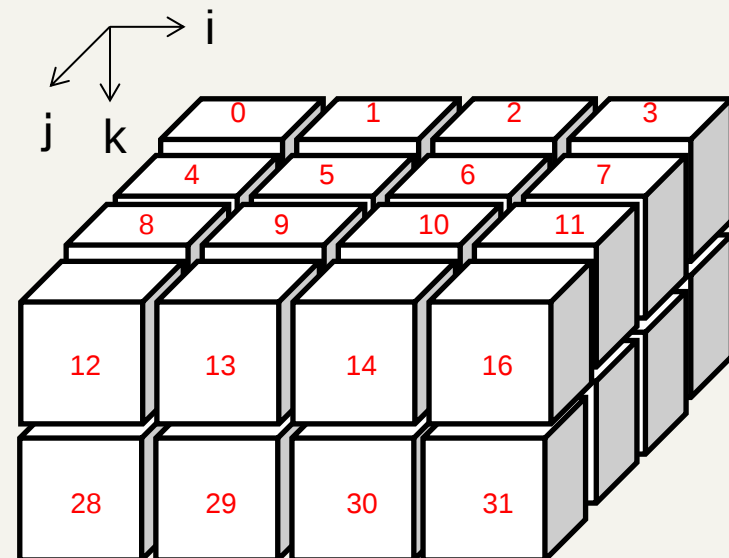
$\dot{u}_p^{n+1/2}$ 粒子速度

$$\sigma_{pq}^{n+1} = \sigma_{pq}^n + \left[\lambda \left(\frac{\partial \dot{u}_x^{n+\frac{1}{2}}}{\partial x} + \frac{\partial \dot{u}_y^{n+\frac{1}{2}}}{\partial y} + \frac{\partial \dot{u}_z^{n+\frac{1}{2}}}{\partial z} \right) \delta_{pq} + \mu \left(\frac{\partial \dot{u}_p^{n+\frac{1}{2}}}{\partial q} + \frac{\partial \dot{u}_q^{n+\frac{1}{2}}}{\partial p} \right) \right] \Delta t, \quad (p, q) = (x, y, z) \quad (7)$$

- ここでは速度と応力が $\Delta t/2$ だけずれた時間で評価されるため、速度と応力が互い違いに時間積分することになる。

1. 弾性波動並列シミュレーション(地震)

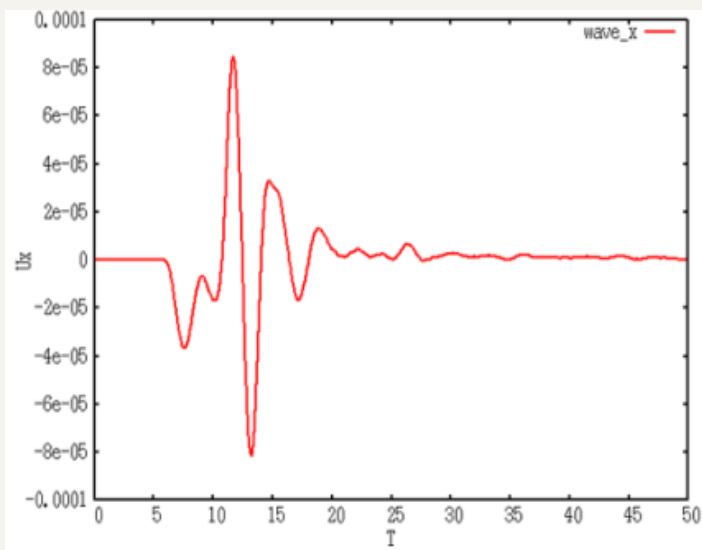
- Staggered グリッド、陽解法
- 3次元/2次元モデル
- 等間隔格子
- 空間微分: 2次、4次、8次精度
- MPI並列は3次元領域分割
- MPI/OpenMPハイブリッド並列



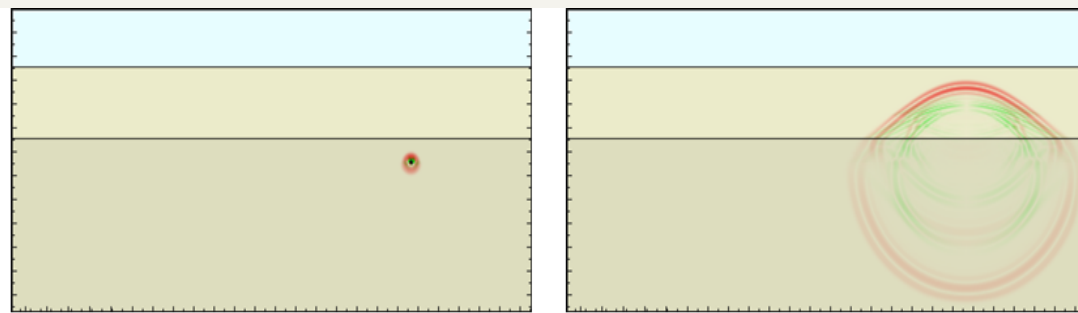
3次元領域分割
(赤文字: MPIランク)

2. サンプルプログラムとインターフェイス

- 観測点の波形
- 波動場のスナップショット



観測点の波形



波動場のスナップショット

ppOpen-APPL/FDM library 概要 (cont.)



- ドキュメント

- ppOpen-APPL/FDM libraryの使い方: ユーザマニュアル
- コード内のモジュール説明

Contents	
1. Outline of seismic_2D/seismic_3D	5
1.1 Parallel simulation of seismic wave propagation in heterogeneous elastic media using ppOpenFDM	5
1.2 Grid and coordinate system	5
1.3 Equations of Motion for 3D Seismic Wavefields	5
1.4 Boundary conditions	6
1.4.1 Absorbing boundary	6
1.4.2 Free surface boundary	7
1.5 Anelastic attenuations	7
1.6 Spatial Differentiation	8
1.7 Input parameters	8
1.8 Requirements of time integration (CFL Condition)	9
1.9 Output and visualization of seismic waves	9
2. Parallel FDM simulation and performance	10
2.1 Domain partitioning and MPI	10
2.2 Parallel programming structure	11
2.3 Performance of parallel FDM simulation	11
2.4 Flat MPI model vs. thread MPI Hybrid model	12
References	13
3. Module/subroutine reference	14
3.1 seismic2d_jav	15
3.1.1 module ppohFDM_m_absorb	17
3.1.2 module ppohFDM_m_convair	18
3.1.3 module ppohFDM_m_kernel	19
3.1.4 module ppohFDM_m_medium	22
3.1.5 module ppohFDM_m_output	23
3.1.6 module ppohFDM_m_params	24
3.1.7 module ppohFDM_m_report	25
3.1.8 module ppohFDM_m_source	27
3.1.9 module ppohFDM_m_stdio	30
3.1.10 module ppohFDM_m_surfbc	39
3.1.11 module ppohFDM_m_match	37
3.2 seismic3d	39
3.2.1 module ppohFDM_boundary	50

3.2.1 module ppohFDM_boundary.

Description

This module applying a zero-stress boundary condition on free surface. Zero stress value is applied to stress components (S_{pz} , $p=x,y,z$) and the results of spatial derivatives just above and below the free-surface boundary are recalculated by using a one-side differentiation scheme. This scheme can treat irregular boundary as well as a flat boundary.

Dependency

use ppohFDM_stdio.
use ppohFDM_param.

subroutine ppohFDM_bc_zero_stress.

(KFSZ, NIFS, NJFS, IFSX, IFSY, IFSZ, JFSX, JFSY, JFSZ).

Description

Applying zero stress value to stress components (S_{pz} , $p=x,y,z$) on free surface.

Arguments

integer, intent(in) :: KFSZ(NXP0:NXP1,NYP0:NYP1) ! depth of the free surface
integer, intent(in) :: NIFS, NJFS ! number of points in x and y directions to examine free surface conditions
integer, intent(in) :: IFSX(NFSMAX), IFSY(NFSMAX), IFSZ(NFSMAX) ! evaluation point of free surface condition in x,y,z
integer, intent(in) :: JFSX(NFSMAX), JFSY(NFSMAX), JFSZ(NFSMAX)

- MITライセンス

- 公開されているコードはユーザが自由に手を加えることができる

ppOpen-APPL/FDM libraryを用いた 弾性波動伝播シミュレーションの計算手順



入力

- 計算パラメータの設定
- ソース、観測点、層構造の設定

計算

出力&可視化

- 観測点における波形
- 波動場

1. 入力パラメータの例 (計算パラメータ)



1. 計算パラメータ (m_param.f90)

モデルサイズ: 128*128*128

格子サイズ: 0.5

時間間隔: 0.025

MPI領域分割: 2*2*2

!-- << Model Size and Grid Width >>

integer, parameter :: NX = 128

integer, parameter :: NY = 128

integer, parameter :: NZ = 128

integer, parameter :: KFS = 25

integer, parameter :: NX1 = NX+1

integer, parameter :: NY1 = NY+1

integer, parameter :: NZ1 = NZ+1

integer, parameter :: NTMAX = 2000

integer, parameter :: NWRITE = 10

real(PN), parameter :: DX = 0.5_PN

real(PN), parameter :: DY = 0.5_PN

real(PN), parameter :: DZ = 0.5_PN

real(PN), parameter :: DT = 0.025_PN

integer, parameter :: NDUMP = 5

モデルサイズ

格子間隔

時間間隔

!--<< Parallel >>

integer, parameter :: IP = 2

integer, parameter :: JP = 2

integer, parameter :: KP = 2

integer, parameter :: NP = IP*JP*KP ! Number of process

integer, parameter :: NL = 4 ! Order of the fd scheme

MPI領域分割

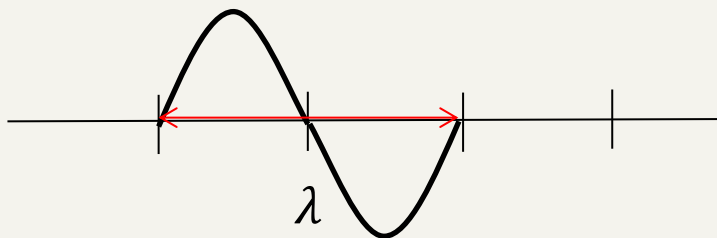
最大周波数と安定条件

(1) 最大周波数の決め方

$$\frac{\lambda^{min}}{\Delta x} = 6$$

4次精度の場合6-8個
(経験的に)

$$\lambda^{min} = 6\Delta x = 6 \times 0.5 = \mathbf{3.0}$$



$$V_s^{min} = f^{max} \times \lambda^{min}$$

(伝わる速度) (周波数) (波長)

$$1.7 = f^{max} \times \mathbf{3.0}$$

$$f^{max} = \mathbf{0.6(Hz)}$$

(2) Δtの決め方

$$\Delta t < 0.2 \frac{\Delta x}{V^{max}}$$

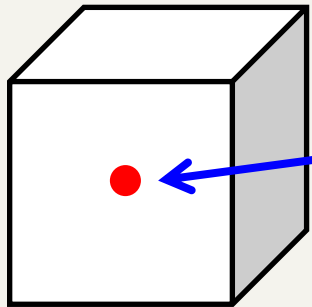
$$\Delta t < 0.2 \frac{0.5}{4.0} = \mathbf{0.025}$$

青: 未知数

黒: 既知数

1. 入力パラメータの例 (ソースの設定、地下構造の設定)

ソースの設定 (source.dat)

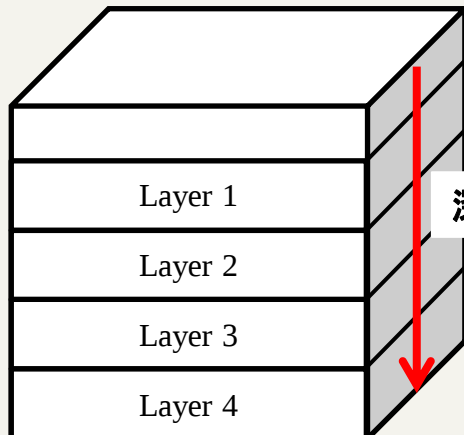


ソース

source.dat

```
16.0 16.0 16.0 # ソースの位置: X(km), Y(km), Z(km)
0.0 45.0 90.0 # 断層パラメータ
1.0 2.0        # ソース時間: at, t0 (s)
```

地下構造の設定 medium.dat



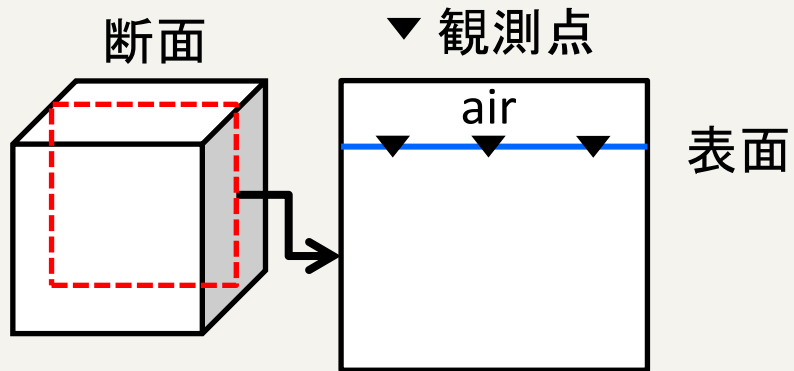
深さ (km)

medium.dat

```
4 # 地下構造の層の数
20 2.3 3.0 1.7 # 深さ(km), 密度 (t/m3),
                P波速度(km/s), S波速度 (km/s)
30 2.3 3.3 2.3
40 2.7 5.0 3.3
50 2.7 6.0 4.0
```

1. 入力パラメータの例 (観測点の設定)

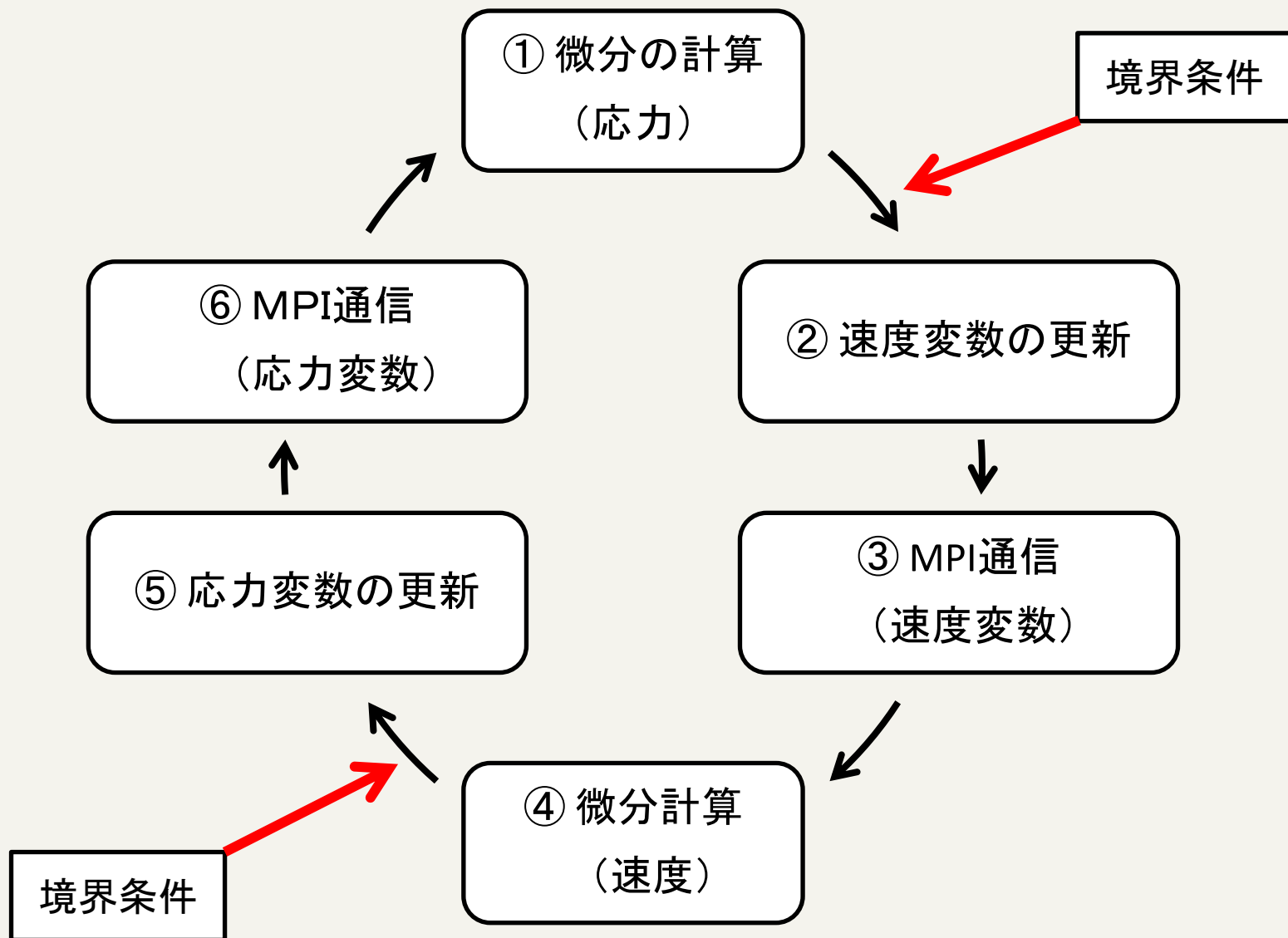
観測点の設定
(station.dat)



station.dat

```
4                                # ステーション数
10.0 10.0 0.0                  # ステーション1: X, Y, Z(km)
                                の設定
40.0 10.0 0.0
10.0 40.0 0.0
40.0 40.0 0.0
```

2. 計算手順



3. 出力および可視化

- 出力データ
 - 各MPIランクで出力されている

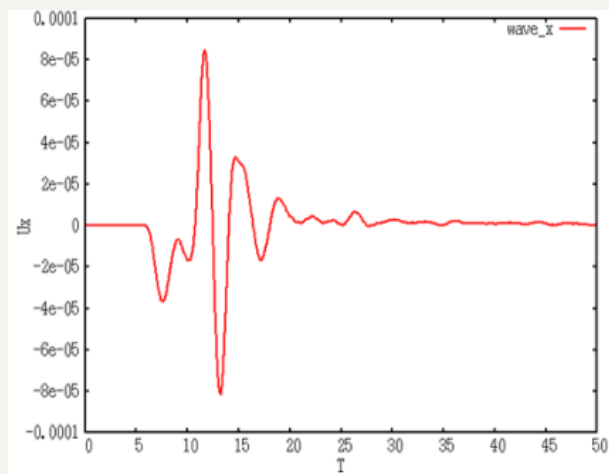
SEISM3D3.prm	SEISM3D3.SUR.000.001.000	SEISM3D3.XY.001.001.000
SEISM3D3.SPS.000.000.000	SEISM3D3.SUR.000.001.001	SEISM3D3.XZ.000.000.000
SEISM3D3.SPS.000.000.001	SEISM3D3.SUR.001.000.000	SEISM3D3.XZ.000.000.001
SEISM3D3.SPS.000.001.000	SEISM3D3.SUR.001.000.001	SEISM3D3.XZ.001.000.000
SEISM3D3.SPS.000.001.001	SEISM3D3.SUR.001.001.000	SEISM3D3.XZ.001.000.001
SEISM3D3.SPS.001.000.000	SEISM3D3.SUR.001.001.001	SEISM3D3.YZ.000.000.000
SEISM3D3.SPS.001.000.001	SEISM3D3.WAV.000.000.000	SEISM3D3.YZ.000.000.001
SEISM3D3.SPS.001.001.000	SEISM3D3.WAV.000.000.001	SEISM3D3.YZ.000.001.000
SEISM3D3.SPS.001.001.001	SEISM3D3.XY.000.000.000	SEISM3D3.YZ.000.001.001
SEISM3D3.SUR.000.000.000	SEISM3D3.XY.000.001.000	
SEISM3D3.SUR.000.000.001	SEISM3D3.XY.001.000.000	

SEISM3D3.prm
SEISM3D3.WAV.***
SEISM3D3.SPS.***
SEISM3D3.SUR.***
SEISM3D3.XY(XZ, YZ).***

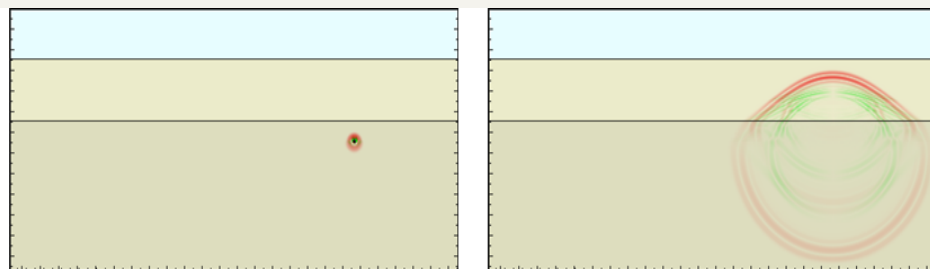
計算パラメータ
観測点における波形
P波とS波の波動場
表面上での波動場
各断面での波動場

3. 出力および可視化 (cont.)

- `./tools/seismic_3D-tools`においてmakeすると4つ実行ファイル(catsnap, catwav, ppmxy3d3, rwav3d)が生成される
- `% catsnap SEISM3D3.prm` → 分割されたファイルが結合される(波動場)
- `% catwav SEISM3D3.prm` → 分割されたファイル が結合される(波形)
- `% ppmxy3d3` → 波動場のスナップショット (xvやimagemagickで可視化)
- `% rwav3d` → 観測点の波形 (gnuplotで可視化)

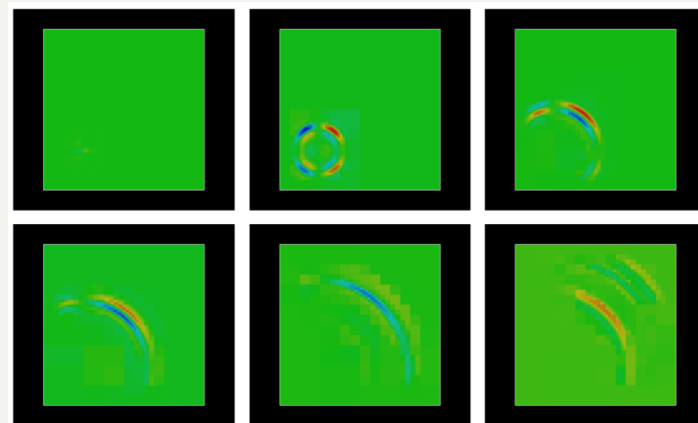
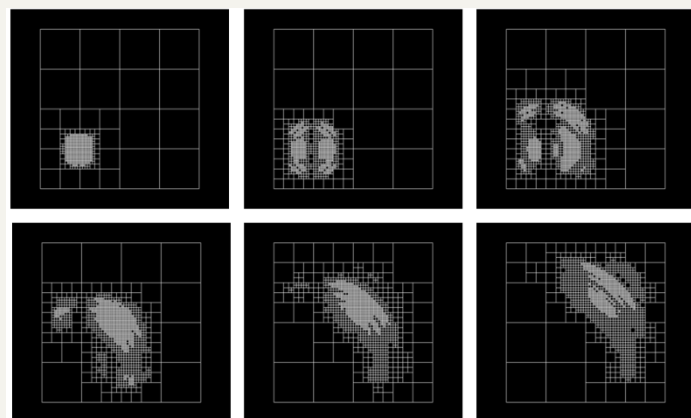


観測点の波形



波動場のスナップショット
(P波: 赤、S波: 緑)

- プロジェクト内で公開している大規模データのための可視化ライブラリppOpen-MATH/VIS library
- ppOpen-APPL/FDM library に実装済み
 - ./src/seismic_3D/1.ppohFDM-ppohVIS
 - 出力ファイルは ./src/seismic_3D/1.ppohFDM-ppohVIS
 - control.datのMaxVoxelCountやMaxRefineLevelの値を大きくすると細くなる (./examples/seismic_3D-example)
 - Paraviewで可視化することができる



VXの速度場

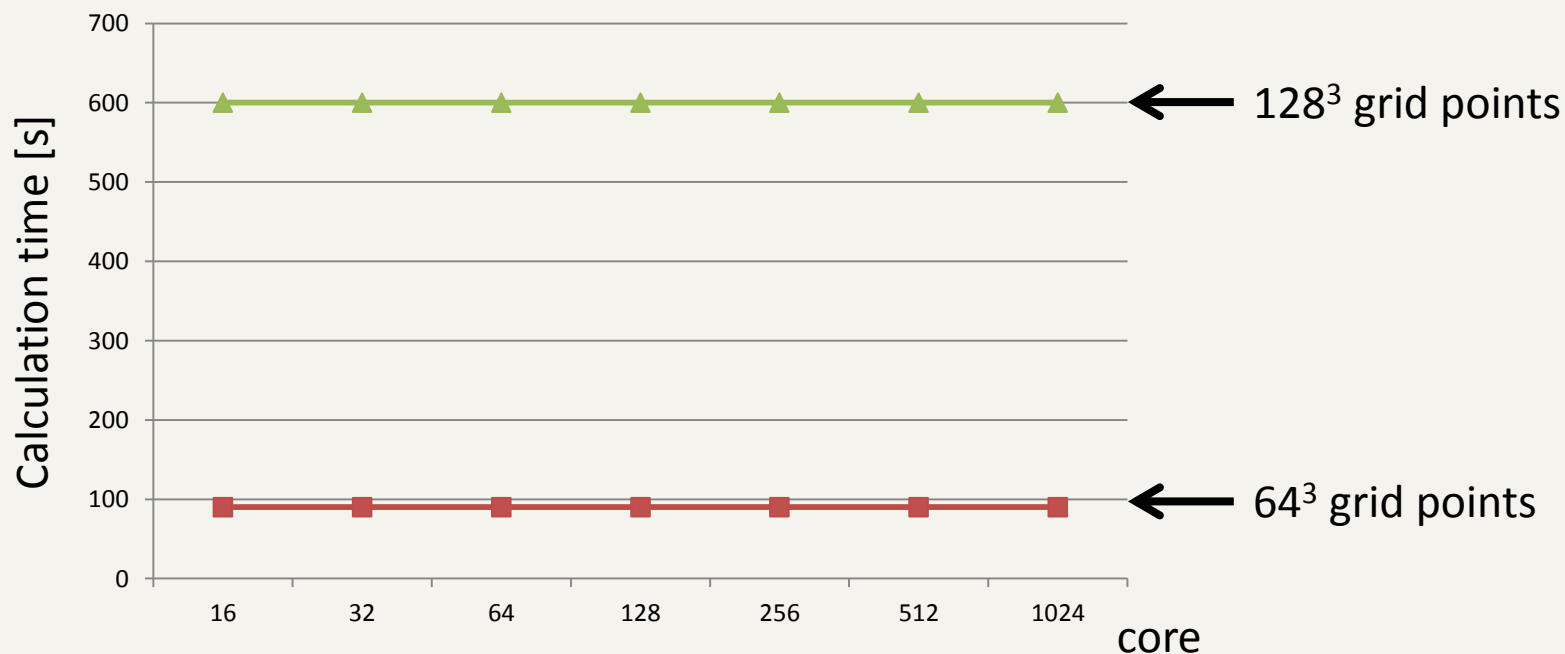
へテロな環境下における Performance Test*

Mori F et al., Lecture Notes in Computer Science (LNCS 8969), pp.66-76, 2015.

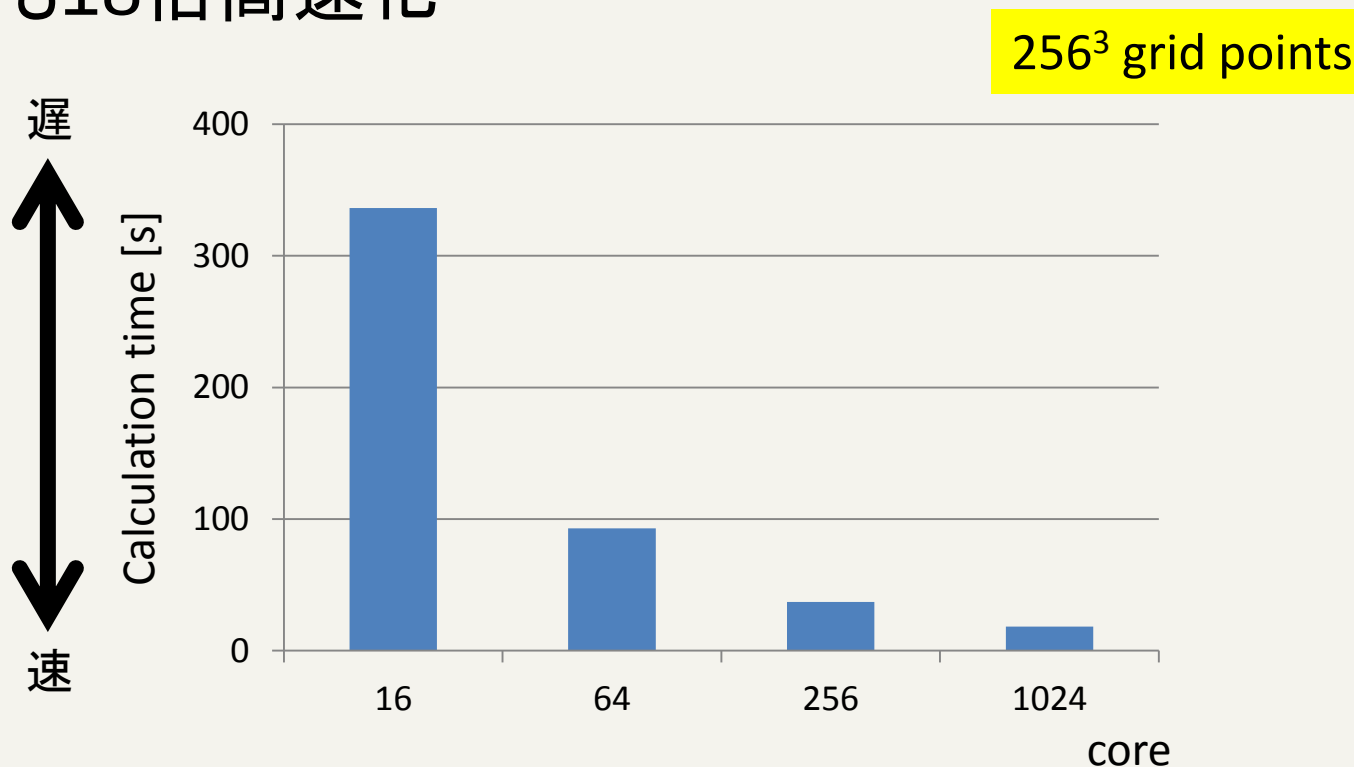
Mori F et al., 11th International Meeting High Performance Computing for Computational Science (VECPAR2014)

Weak scaling test in flat MPI

- モデルサイズ: 64^3 , 128^3 grid points
- 並列数: 16~1024 コア
- 3D domain partitioning of MPI



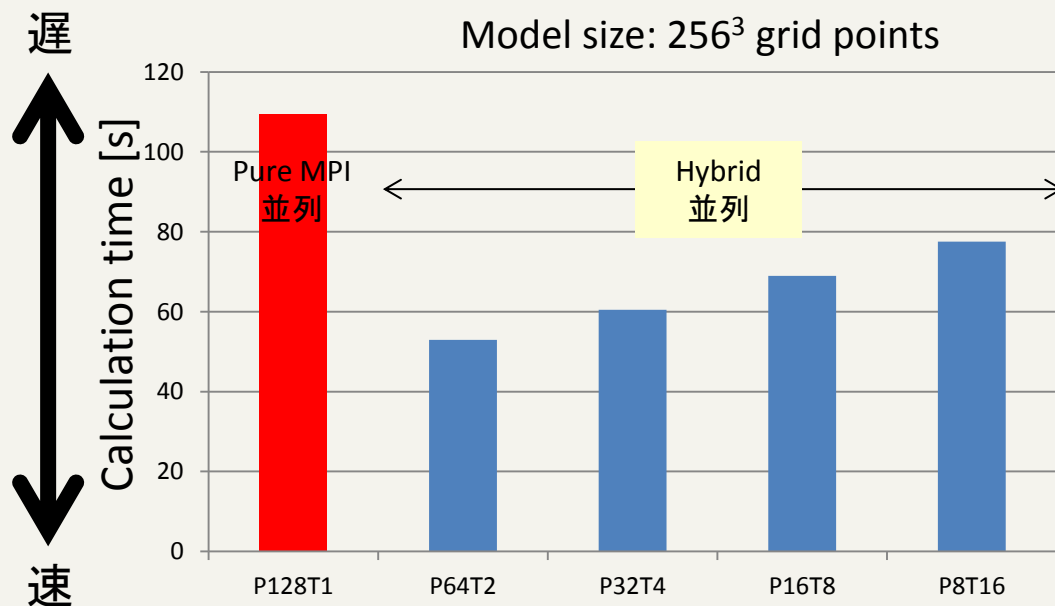
- 計算時間はコア数が増加することで低減している
 - 1024コアでの計算時間は、16コアの計算時間よりも18倍高速化



Storing scaling test in the MPI/OpenMP hybrid parallel computing on FX10



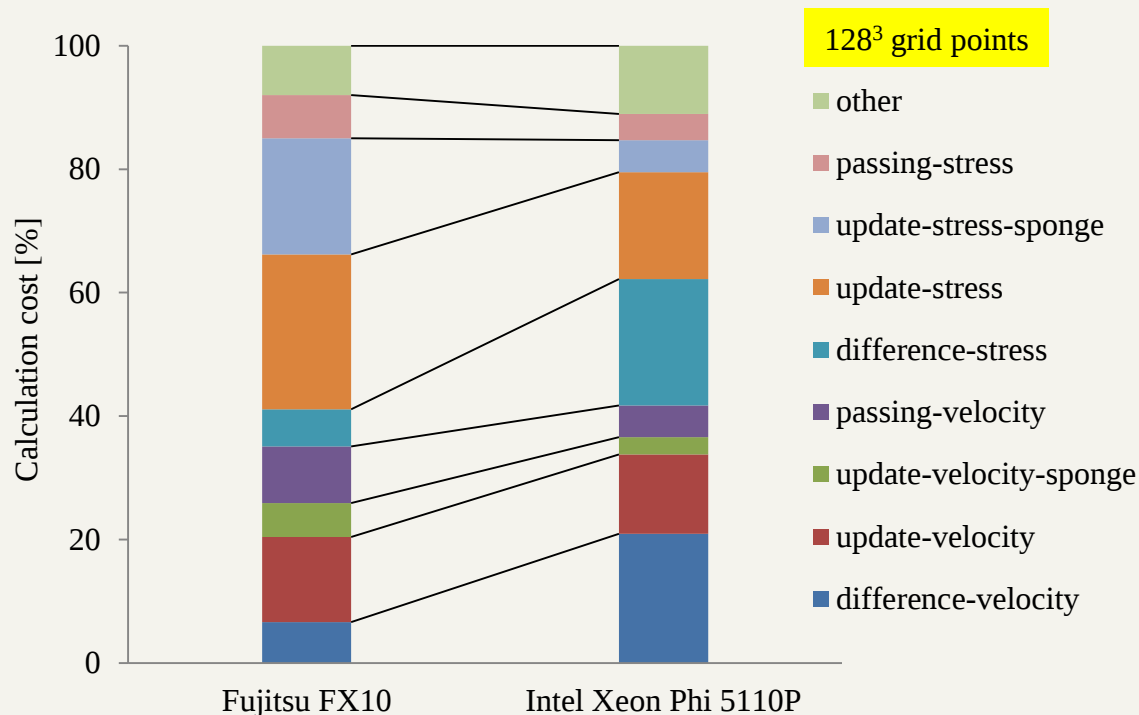
- パフォーマンステストは8ノード(128コア, 16コア/ノード)を使って評価
 - P128T1, P64T2, P32T4, P16T8, P8T16 (Pはプロセス、Tはスレッド)
- モデルサイズ
 - 256³ grid points
- 最小の計算時間は、P64T2 による並列のときであった
 - Pure MPI並列よりも **P64T2** の方が2倍高速化していた



ヘテロな環境下でのパフォーマンステスト

	Intel Xeon Phi 5110P @地震研	Fujitsu FX10 @東大情報セ	Xeon CPU (E5-4627 v2) @地震研
CPU clocks	1.053 GHz	1.848 GHz	3.3 GHz
Number of Cores	60	16	8
Thread/core	4	1	1
Size of L2 cache	512KB	12MB	2 MB
Peak Performance	1.01 TFLOPS	236.5 GFLOPS	211.2 GFLOPS
Peak Memory Bandwidth	320 GB/sec	85 GB/sec	59.7 GB/sec

- 公開中の弾性波動伝播コードにおいて、Fujitsu FX10やXeon Phiを用いた際、計算コストが高いカーネル
 - 応力更新
 - 速度更新

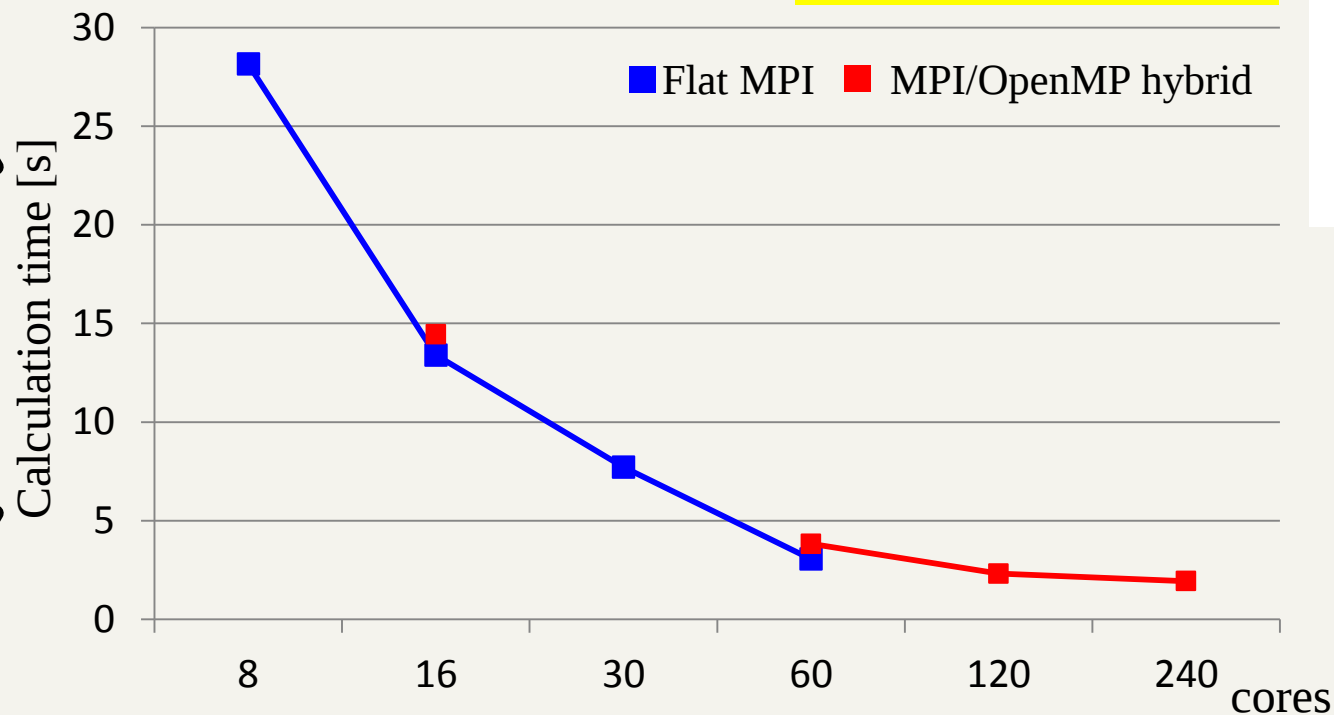


FDM calculation using 16 MPI parallel processes in 128³ grid points

Flat MPI processing and MPI/OpenMP hybrid parallel computing in ppOpen-APPL/FDM

遅
↑
↓
速

Update-velocity kernel
240³ grid points



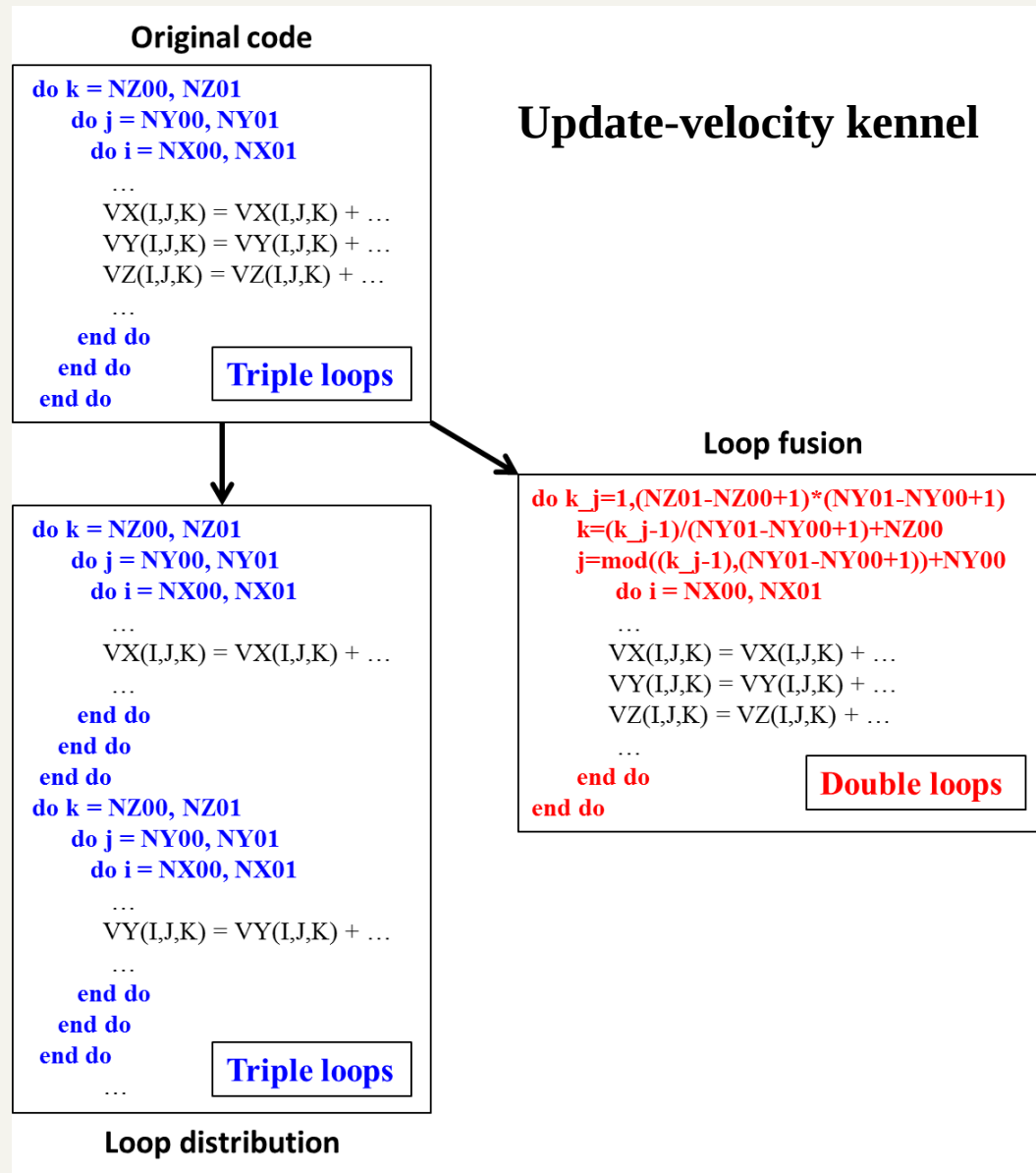
<http://www.intel.co.jp/>

Xeon Phi のNative modeで実行

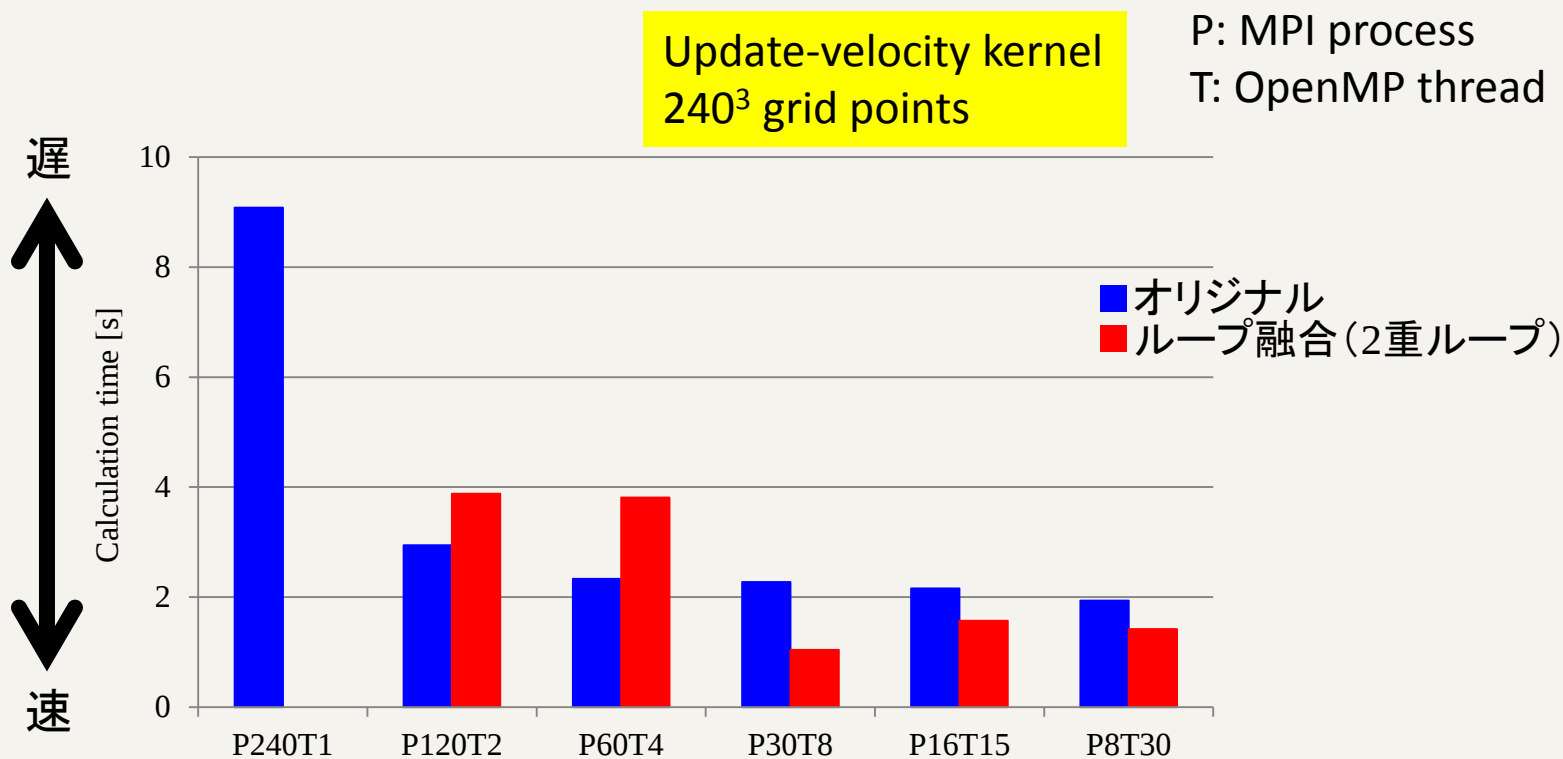
- 60コア(物理コア)までは、Flat MPI並列の方がMPI/OpenMP ハイブリッド並列より高速
- 240コア(Hyper-Threading function)を使うことによって、MPI/OpenMPハイブリッド並列は、Flat MPI並列(60コア)より更に1.57倍速度向上できた
- Update-stress kernelも同様である

Performance of kernel loop optimization

- ループの最適化
 - ループ融合
(Loop fusion)
 - ループ分割
(Loop distribution)



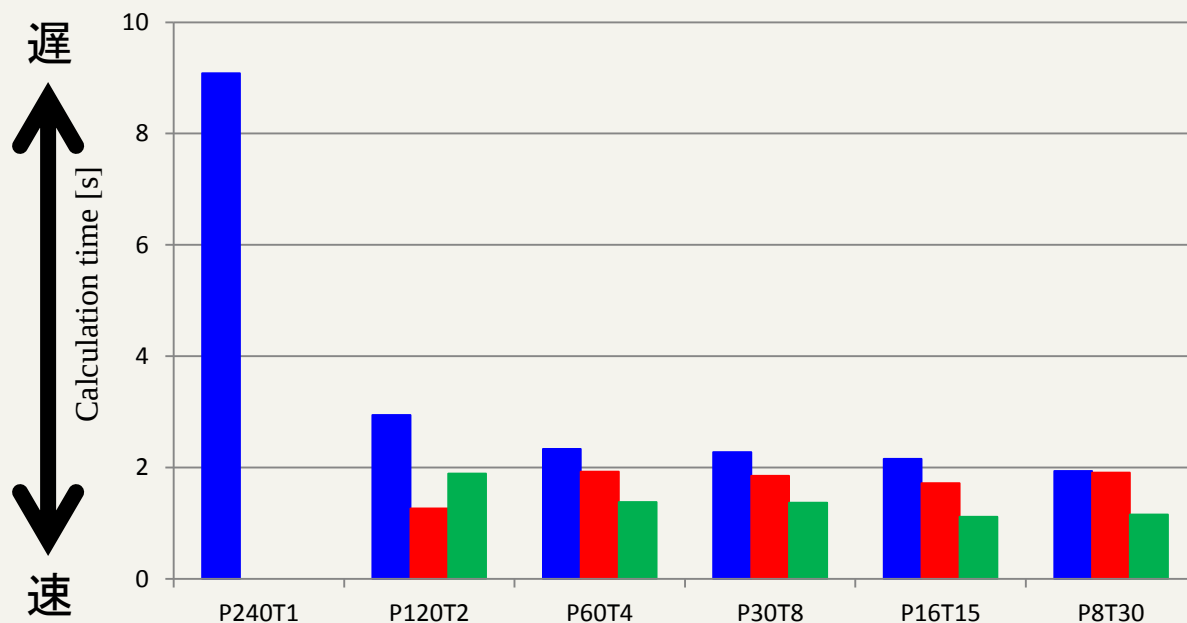
Performance of kernel loop optimization (Loop fusion) in **Triple loops** and **Double loops**



ループ融合によるパフォーマンス (240コアを使用した場合)

- スレッド並列数が8スレッド以上の場合、**3重ループ**より**2重ループ**の方が2.2倍高速化 (P30T8)
- 1重ループへループ融合したが、3重ループより1重ループの方が8.0倍低速化 (P8T30)
- Update-stress kernelにおいても傾向は同じ

Performance of kernel loop optimization (Loop distribution) in **Triple loops** and **Double loops**



Update-velocity kernel
240³ grid points

P: MPI process

T: OpenMP thread

■ オリジナル

■ ループ分割 (3つのループ)

■ ループ分割 (3つのループ
+ K, Jをループ融合)

ループ分割によるパフォーマンス (240コアを使用した場合)

- スレッド並列数が4スレッド以上の場合、**オリジナル (ループ分割なし)** より最適化した方が高速化し、特に**ループ分割 (3つのループ + K, Jをループ融合)** の方は1.9倍高速化した (P16T15)
- Update-stress kernelも同様である

- 並列計算においてデータ転送能力(Byte/s)と演算性能(Flops/s)の関係が重要である
- CPUの性能向上によって近年のスパコンにおけるB/F値は徐々に減少傾向にある
ES(B/F値4.0)→ES2 (B/F値2.5)→京(B/F値0.5)
- 高B/F値のコードは、次世代のスパコンで高い並列性能を得ることが難しい
→ 要求B/F値の低減が必要になる

Procedure of decreasing B/F ratio

Originalコード

```
do K=1,NZ
  do J=1,NY
    do I=1,NX
      ...
      DxSxx(I,J,K)=Sxx(I,J,K)-Sxx(I-1,J,K)
                    -Sxx(I+1,J,K)-Sxx(I-2,J,K)
    end do
  end do
end do
```

**Calculate Derivatives
kernel**

```
do K=Nz00,Nz01
  do J=Ny00,Ny01
    do I=Nx00,Nx01
      ...
      Vx(I,J,K)=Vx(I,J,K)+(DxSxx(I,J,K)+...
      Vy(I,J,K)=Vy(I,J,K)+(DxSxy(I,J,K)+...
      Vz(I,J,K)=Vz(I,J,K)+(DxSxz(I,J,K)+...
    end do
  end do
end do
```

**Update-velocity
kernel**

(update-velocity)要求B/F=2.7, (update-stress)要求B/F=1.7

Modifiedコード (Calculate Derivatives kernel + Update-velocity kernel)

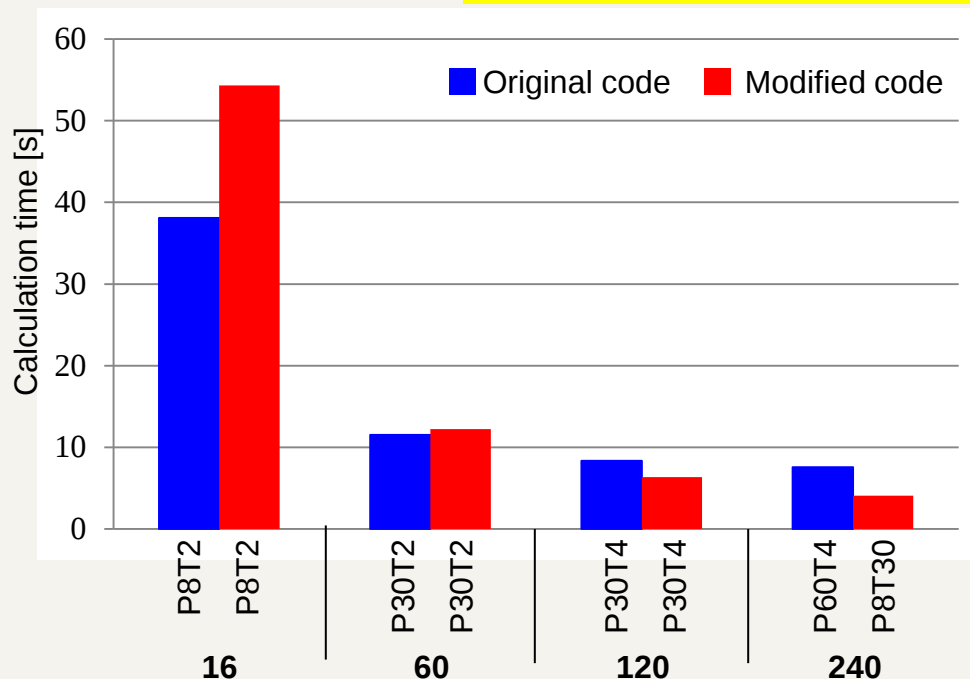
```
do K=NZ00, NZ01
  do J=NY00, NY01
    do I=NX00, NX01
      ! Calculate Derivatives kernel
      DxSxx0=(Sxx(I+1,J,K)-Sxx(I-1,J,K))*C40/dx-(Sxx(I+2,J,K)-Sxx(I-1,J,K))*C41/dx
      DxSxy0=(Sxy(I,J,K)-Sxy(I-1,J,K))*C40/dx-(Sxy(I+1,J,K)-Sxy(I-2,J,K))*C41/dx
      DxSxz0=(Sxz(I,J,K)-Sxz(I-1,J,K))*C40/dx-(Sxz(I+1,J,K)-Sxz(I-2,J,K))*C41/dx
      ...
      ! Update-velocity kernel
      Vx(I,J,K)=Vx(I,J,K)+(DxSxx0+DySxy0+DzSxz0)*ROX*DT
      Vy(I,J,K)=Vy(I,J,K)+(DxSxy0+DySyy0+DzSyz0)*ROY*DT
      Vz(I,J,K)=Vz(I,J,K)+(DxSxz0+DySyz0+DzSzz0)*ROZ*DT
    end do
  end do
end do
```

(update-velocity)要求B/F=0.4, (update-stress)要求B/F=0.4

Xeon Phi 5110P @ERI

Update-velocity kernel
240³ grid points

P: MPI process numbers
T: OpenMP thread numbers



- MPI/OpenMP Hybrid 並列において、**Modified code**が**Original code**より高速になるのはHyper Threadingを使用したときであり、P8T30において1.9倍高速化がみられた
- Update-stress kernelにおいても同傾向である

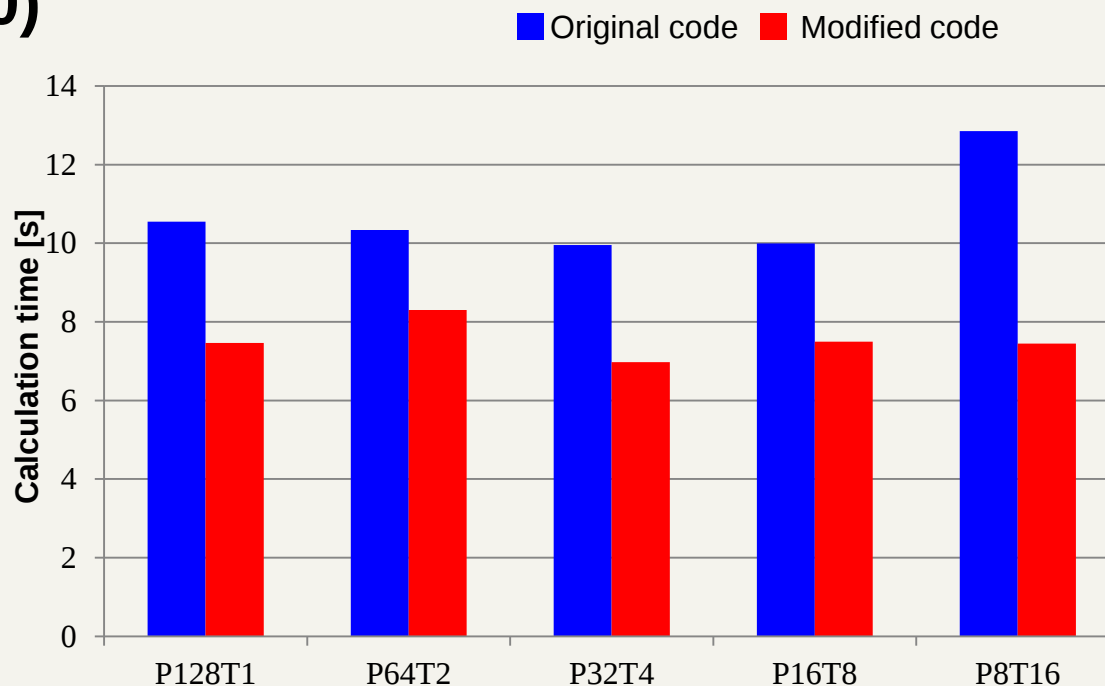
Original code and Modified code using MPI/OpenMP hybrid parallel computing in Fujitsu FX10



**CPU: Sparc 64 lxfx
@ITC(FX10)**

Update-velocity kernel
512³ grid points

P: MPI process
T: OpenMP thread



FX10におけるOriginal code と Modified codeの比較

- **Original code**の最速であるP32T4において**Modified code**は1.4倍高速化がみられた
- Update-stress kernelにおいても同傾向である

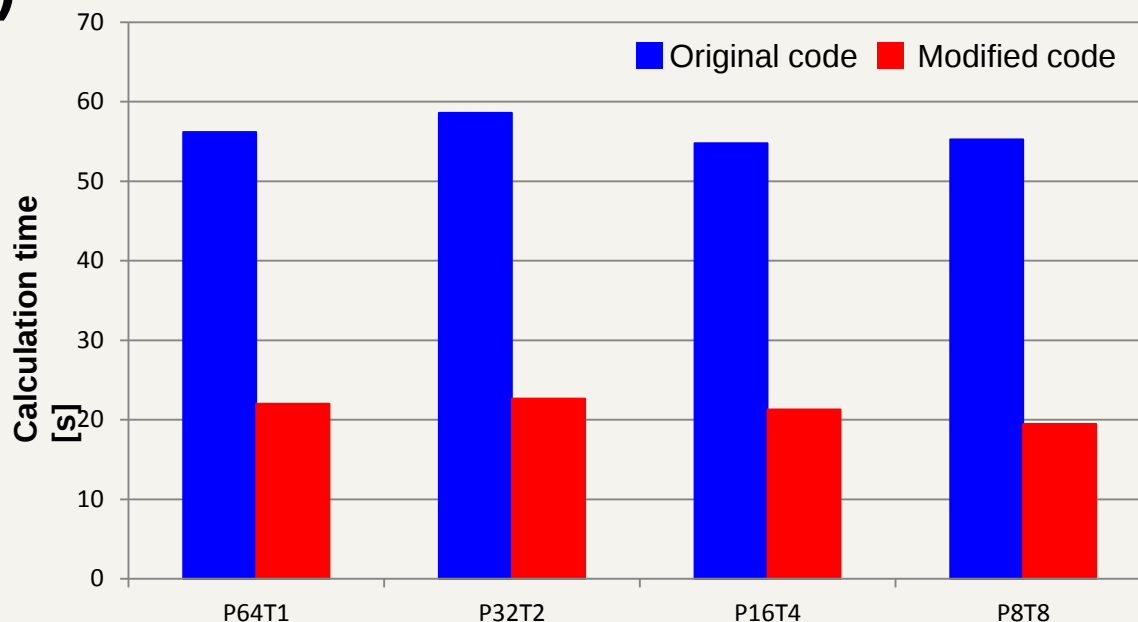
Original code and Modified code using MPI/OpenMP hybrid parallel computing in Intel Xeon CPU Cluster



**CPU: Intel Xeon E5-4627 v2
@ERI(EIC)**

Update-velocity kernel
256³ grid points

P: MPI process
T: OpenMP thread



Intel Xeon CPU Clusterにおける**Original code**と**Modified code**の比較

- **Original code**の最速であるP16T4において**Modified code(P8T8)**は2.8倍高速化がみられた
- Update-stress kernelにおいても同傾向である

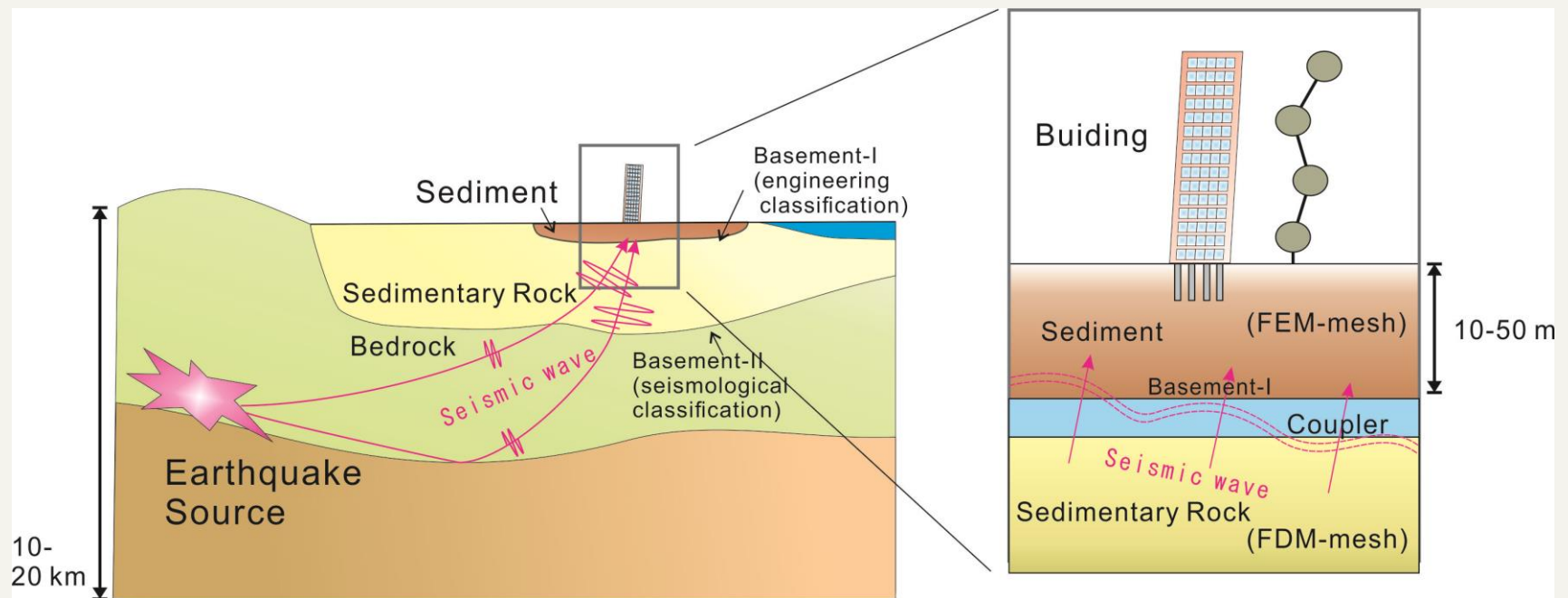
- Pure MPI並列 vs Hybrid並列
 - Hybrid並列の方が有効的である
- Intel Xeon Phi Co-processorの最適化
 - チューニングとしてループ分割が全体のパフォーマンスを考えたとき有効的である
 - スレッド数の増加に対して、ループ分割＋ループ融合が安定的に効果が見込める
- 要求B/F値の低減
 - ヘテロな環境においてメモリ律速から緩和させることによってCPU性能が顕在化した可能性がある
 - モデルサイズの違いがあることが推測される

ppOpen-APPL/FDMの実例 大規模連成計算**

Matsumoto M et al., Multi-scale Coupling Simulation of Seismic Waves and Building Vibrations Using ppOpen-HPC,
Procedia Computer Science 51: 1514-1523 (2015)

大規模連成計算の取り組み

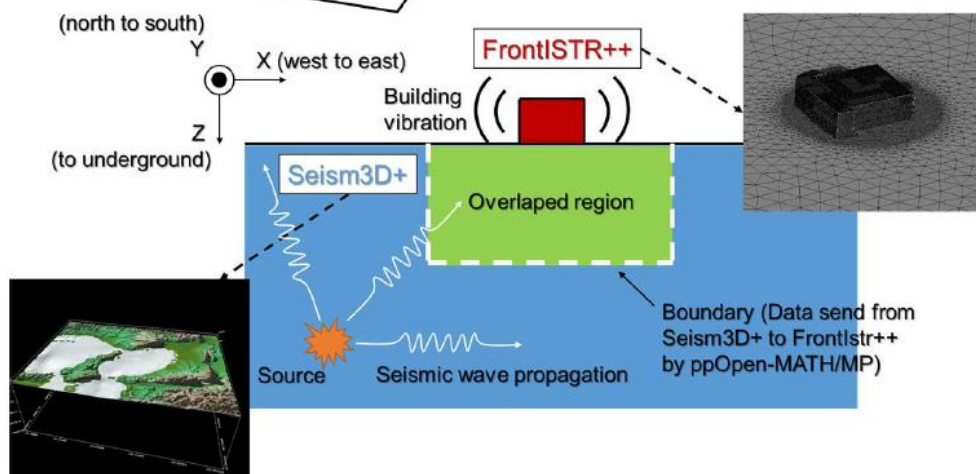
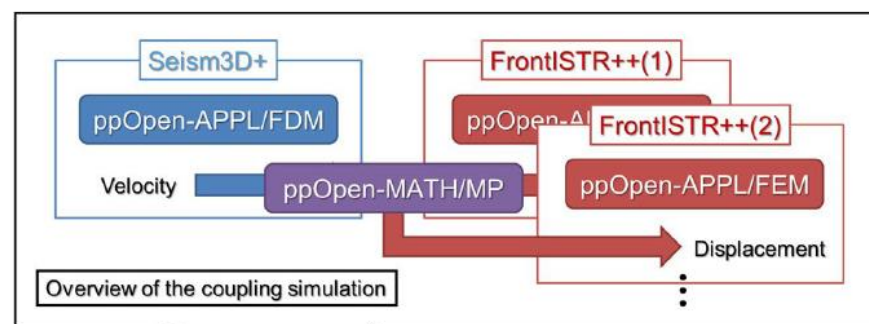
- ppOpen-HPCプロジェクト内で開発されているppOpen-MATH/MP couplerを用いて差分法(地震動→ ppOpen-APPL/FDM)と有限要素法(構造解析→ ppOpen-APPL/FEM)を連成し、大規模連成計算をおこなっている



FDM: Seismic Wave Propagation → ppOpen-MATH/MP → FEM: Building Response

Overview of Coupling simulation

- **淡路**を震源とした地震を想定し、ポートアイランドにある**計算科学研究機構 京コンピュータの建屋**への影響をシミュレーションをおこなった

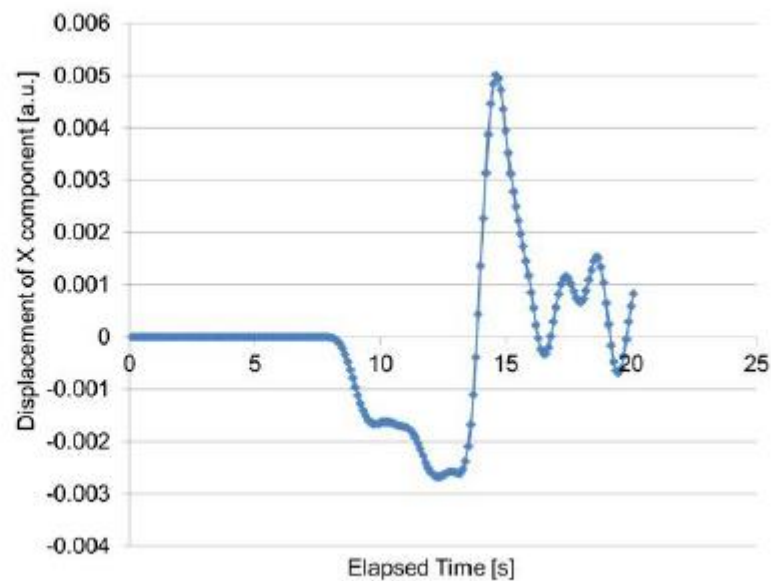
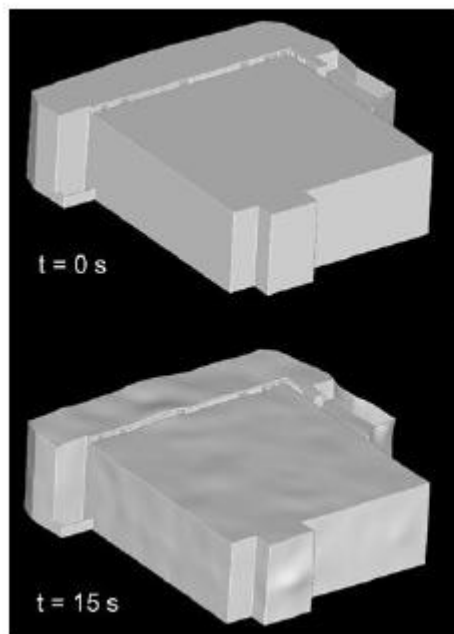


大規模連成解析の解析例

地震動



構造解析



ppOpen-APPL/FDM library

利用方法と演習

はじめに



/home/c31003/share/lecture.tar.gzがありますのでコピーし解凍してください

* 以下のアプリケーションをインストールしてください

Cygwin, Putty (in windows)

Paraview

WinSCP (in windows)

FUJITSU Software Development

ディレクトリとファイルを確認

./lecture以下を確認 (以下このディレクトリをルートディレクトリとして説明)

演習で使うコードは、./lecture/src/seismic_3D以下にあります

```
[c31003@oakleaf-fx-3 lecture]$ ls
doc                LICENSE_ppohAPPL-FDM  ppohMATH-VIS-install  tools
etc                LICENSE_ppohMATH-VIS  ppohMATH-VIS-lib
examples           Makefile              README_ppoh-APPL-FDM
INSTALL_ppohAPPL-FDM Makefile.in           src
[c31003@oakleaf-fx-3 lecture]$ cd src/
[c31003@oakleaf-fx-3 src]$ ls
seismic_2D  seismic_3D
[c31003@oakleaf-fx-3 src]$ cd seismic_3D/
[c31003@oakleaf-fx-3 seismic_3D]$ ls
1. ppohFDM-ppohVIS  2. parallel  3. parallel-opt
```

ppohVISライブラリが実装

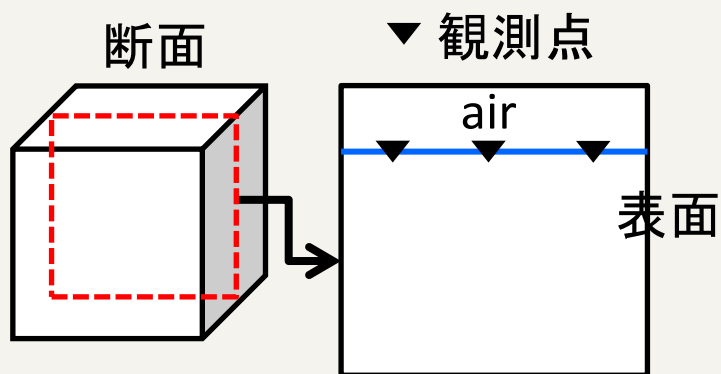
並列コード

並列コード(最適化)

入力パラメータ設定(1)

• 3つの入力パラメータファイル

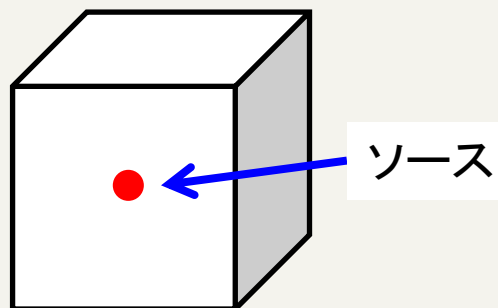
1. 観測点 (X, Y, Z (km))の設定



station.dat

```
4 # ステーション数
10.0 10.0 0.0 # ステーション1: X, Y, Z(km)を設定
40.0 10.0 0.0
10.0 40.0 0.0
40.0 40.0 0.0
```

2. ソースパラメータを設定



source.dat

```
16.0 16.0 16.0 # ソースの位置: X(km), Y(km), Z(km)
0.0 45.0 90.0 # 断層パラメータ
1.0 2.0 # ソース時間: at, t0 (s)
```

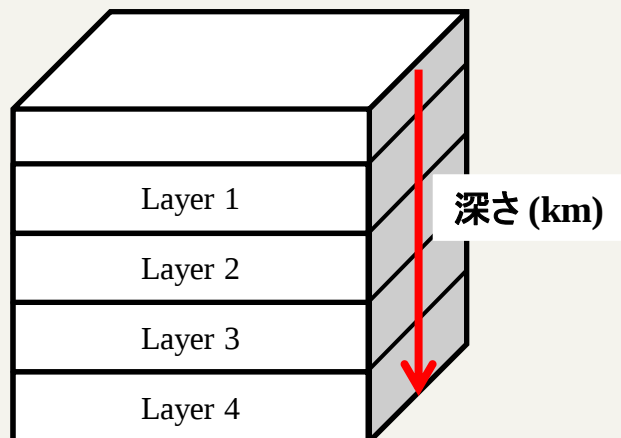
3. 地下構造モデル(深さ(depth), 密度(R0), P波の速度(VP), S波の速度(VS))を設定

- medium.dat

medium.dat

```
4          # 地下構造の層の数
20 2.3 3.0 1.7  # 深さ(km), 密度 (t/m3),
                # P波速度(km/s), S波速度 (km/s)

30 2.3 3.3 2.3
40 2.7 5.0 3.3
50 2.7 6.0 4.0
```



計算パラメータ (m_param.f90)

(計算モデルとタイムステップ)

- モデルサイズ: NX, NY, NZ
- 格子間隔: DX, DY, DZ
- タイムステップ: NTMAX
- 時間間隔: DT

(MPI: 3次元分割)

- 分割: IP, JP, KP
- プロセス数: NP

```
!-- << Model Size and Grid Width >>
```

```
integer, parameter :: NX = 128
```

```
integer, parameter :: NY = 128
```

```
integer, parameter :: NZ = 128
```

```
integer, parameter :: KFS = 25
```

```
integer, parameter :: NX1 = NX+1
```

```
integer, parameter :: NY1 = NY+1
```

```
integer, parameter :: NZ1 = NZ+1
```

```
integer, parameter :: NTMAX = 1000
```

```
integer, parameter :: NWRITE = 10
```

```
real(PN), parameter :: DX = 0.5_PN
```

```
real(PN), parameter :: DY = 0.5_PN
```

```
real(PN), parameter :: DZ = 0.5_PN
```

```
real(PN), parameter :: DT = 0.025_PN
```

```
integer, parameter :: NDUMP = 5
```

モデルサイズ

タイムステップ

格子間隔

時間間隔

```
!--<< Parallel >>
```

```
integer, parameter :: IP = 2
```

```
integer, parameter :: JP = 2
```

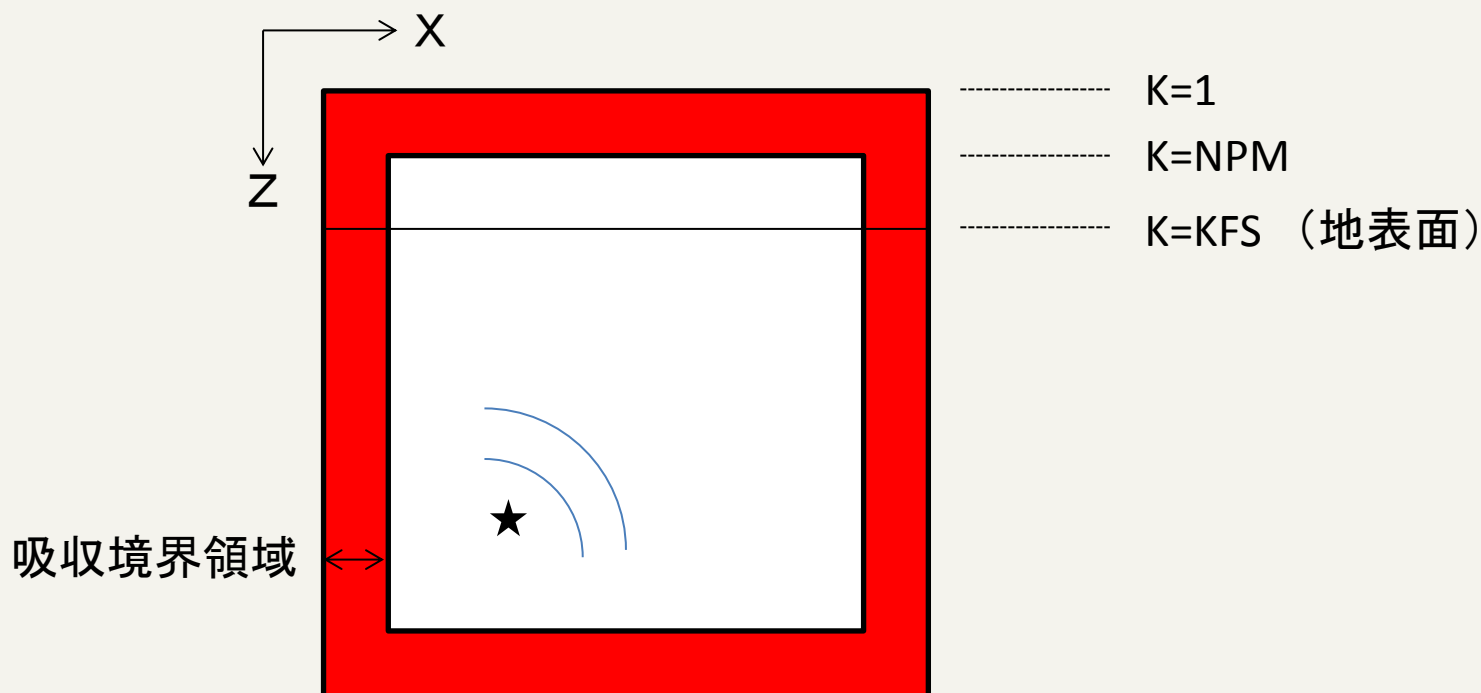
```
integer, parameter :: KP = 2
```

```
integer, parameter :: NP = IP*JP*KP ! Number of process
```

```
integer, parameter :: NL = 4 ! Order of the fd scheme
```

MPI領域分割

計算パラメータの注意点



本コードでは、吸収境界条件を設定しています。吸収境界は`m_param.f90`内の`NPM`変数において、20 gridsに設定しています。また、地表面のリファレンスとして`KFS`変数を定義しています。サンプルでは、`KFS`は5gridsとしています。

1. 媒質の物理値を変化させ、波動場の観測せよ。パラメータは以下の通り

- 計算パラメータの設定 (m_param.f90)
 - モデルサイズ(NX*NY*NZ): 128*128*128
- 媒質の設定 (medium.dat)

```
4
20 2.3 3.0 1.7
30 2.3 3.3 2.3
40 2.7 5.0 3.3
50 2.7 6.0 4.0
```

デフォルト



```
5
10 2.3 2.8 1.2
20 2.3 3.0 1.7
30 2.3 3.3 2.3
40 2.7 5.0 3.3
50 2.7 6.0 4.0
```

- ソースの設定 (source.dat)

```
16.0 16.0 16.0
0.0 45.0 90.0
1.0 2.0
```

デフォルト



```
16.0 16.0 10.0
0.0 45.0 90.0
1.0 2.0
```

演習(1)(cont.)

– 観測点の設定(station.dat)

```
4
10.0 10.0 0.0
40.0 10.0 0.0
10.0 40.0 0.0
40.0 40.0 0.0
```

– control.dat

```
[Refine]
AvailableMemory = 8.0
MaxVoxelCount = 5000
MaxRefineLevel = 1000
[Simple]
ReductionRate = 0.0
```

- ラベルを設定する(seism3d3n.f90のL107)
- 出力間隔を設定する(seism3d3n.f90のL358)
- 出力パラメータの設定(seism3d3n.f90のL375)

• 上記のパラメータとデフォルトパラメータとの違いを確認せよ

1. 媒質のパラメータを変更したときの違い
2. ソースのパラメータを変更したときの違い
3. control.datの**MaxVoxelCount**と**MaxRefineLevel**の値を変更したときの違い確認せよ

演習(1)(cont.)

- コンパイル
 1. % make
 2. % make install
 3. % make seism3d-ppohVIS
 4. % make install

- 実行

- % cd ./src/seismic_3D/1.ppohFDM-ppohVIS
- %pjsub job-lecture-pureMPI

(注意)

Makefile.inにMATH/VISライブラリのインストール先が書かれている。各ユーザはインストール先のディレクトリを場所を絶対パスで書く

```
# Install directory
#PREFIX must be specified as ABSOLUTE/FULL path
PREFIX      = /home/(username)/lecture/ppohHPC
BINDIR      = $(PREFIX)/bin
INCDIR      = $(PREFIX)/include
LIBDIR      = $(PREFIX)/lib
```

Makefile.in

```
#!/bin/sh
                                     job-lecture-pureMPI
                                     ファイルの中身
#PJM -L "rscgrp=tutorial"
#PJM -L "node=1"
#PJM -L "elapse=00:15:00"
#PJM --mpi "proc=8" ←
mpiexec ./seism3d3n
```

m_param.f90で設定したNP(=IP*JP*KP)とproc数を
同じにする

tutorialでは、12ノード(192コア)まで計算が投げられる

- ppOpen-MATH/VISで出力されたinpファイルをparaviewを使って可視化
 - ./src/seismic_3D/1.ppohFDM-ppohVIS/ppohVISにデータが出力されていることを確認
- 自分のPCにデータを移動し、Paraviewを使って可視化する

1. 並列数とモデルサイズを変更して計算時間の変化を計測せよ

- m_param.f90
 - モデルサイズ: 256*256*256 grid points
 - NP: 8, 16, 32, 64, 128
- プロファイル情報を取得するために、計測した部分に call fapp_start, call fapp_stop で挟む
 - seism3d3n.f90 に書き込み済みなので確認すること

!! Velocity Update

```
call fapp_start("region2",1,1)
```

```
call ppohFDM_update_vel ( 1, NXP, 1, NYP, 1, NZP )
```

```
call fapp_stop("region2",1,1)
```

- ./src/seismic_3D/2.parallelのMakefile.optionがpureMPIのコンパイルオプションになっているか確認する

```
# Makefile option setting
# for Intel Fortran compiler
# FC = mpif90
# FFLAGS = -O3                # pure MPI processing
# FFLAGS = -O3 -openmp        # MPI/OpenMP processing

# for Intel Fortran compiler on MIC
# FC = mpif90
# FFLAGS = -O3 -mmic -align array64byte # pure MPI processing
# FFLAGS = -O3 -mmic -openmp -align array64byte # MPI/OpenMP processing

# for FX10 fujitsu compiler
FC = mpifrtpx
FFLAGS = -O3 -Kfast # pure MPI processing
# FFLAGS = -O3 -Kfast -Kopenmp # MPI/OpenMP hybrid

# for Gfortran
# FC = gfortran
# FFLAGS = -O3 -funroll-loops -msse3 -m64
```

Makefile.option
の中身

- Job ファイル

```
#!/bin/sh

#PJM -L "rscgrp=tutorial"
#PJM -L "node=1"
#PJM -L "elapsed=00:15:00"
#PJM --mpi "proc=8"

fapp -C -d prof -L 1 -lhwmm -Hevent=Statistics mpiexec lpgparm -p 256MB -d 256MB -h 256MB -s 256MB
-t 256MB ./seism3d3n
```

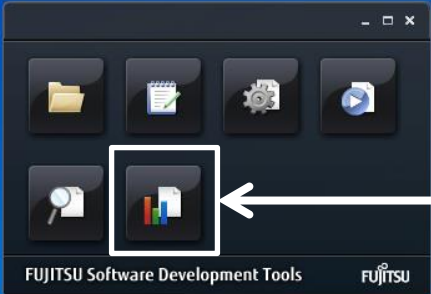
8 pureMPI並列の場合

- プロファイルを保存するディレクトリが必要でprofディレクトリは空にしておく
 - ./src/seismic_3D/2.parallel/prof

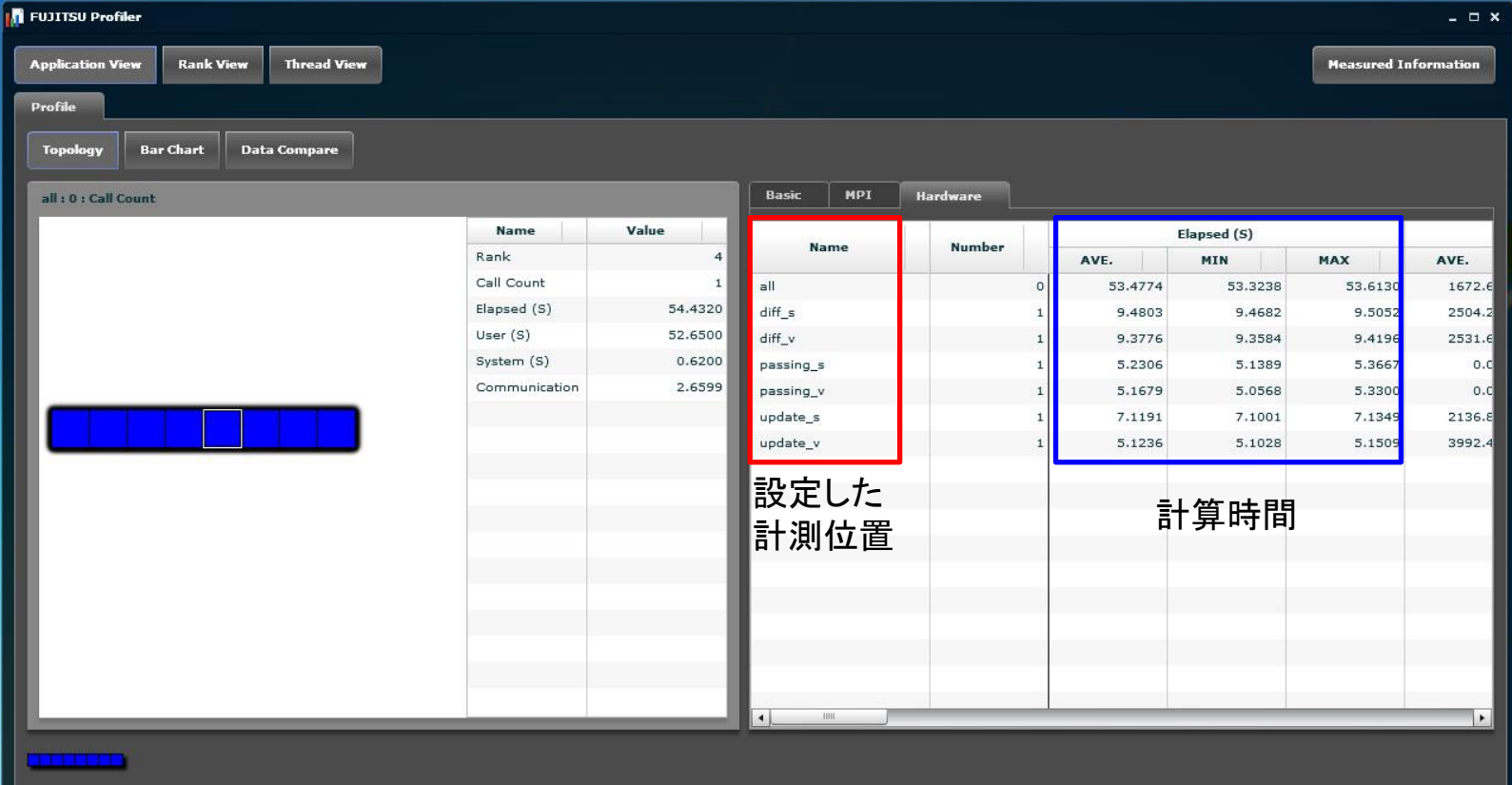
(注意)

計測にあたりseism3d3n.f90のL175, L316にあるI/Oに関連するサブルーチンをコメントアウトを確認

Profile情報(FUJITSU software developer)



プロフィール情報



Application View Rank View Thread View Measured Information

Profile

Topology Bar Chart Data Compare

all : 0 : Call Count

Name	Value
Rank	4
Call Count	1
Elapsed (S)	54.4320
User (S)	52.6500
System (S)	0.6200
Communication	2.6599

Basic MPI Hardware

Name	Number	Elapsed (S)			
		AVE.	MIN	MAX	AVE.
all	0	53.4774	53.3238	53.6130	1672.6
diff_s	1	9.4803	9.4682	9.5052	2504.2
diff_v	1	9.3776	9.3584	9.4196	2531.6
passing_s	1	5.2306	5.1389	5.3667	0.0
passing_v	1	5.1679	5.0568	5.3300	0.0
update_s	1	7.1191	7.1001	7.1349	2136.8
update_v	1	5.1236	5.1028	5.1509	3992.4

設定した計測位置

計算時間

- PureMPI並列、Hybrid並列に計測サブルーチンを実装
- 計測したい部分をcall stopwatch__strt(*)とcall stopwatch__stop(*)で挟む。*は、同じ数字
 - seism3d3n.f90に書き込み済みなので確認すること

```
!! Velocity Update
call stopwatch__strt(2)
call fapp_start("update_v", 1, 1)
call ppohFDM_update_vel ( 1, NXP, 1, NYP, 1, NZP )
call fapp_stop("update_v", 1, 1)
call stopwatch__stop(2)
```

- 標準出力部分に計測結果が出力
 - job.o***を確認

```
stopwatch: 0 157.611359
stopwatch: 1 32.8303986
stopwatch: 2 17.1797485
stopwatch: 3 9.51004028
stopwatch: 4 33.3803864
stopwatch: 5 25.3695374
stopwatch: 6 9.75003052
```

演習(3) Hybrid parallel computing



- ./src/seismic_3D/2.parallelのMakefile.optionがhybridのコンパイルオプションになっているか確認する

```
# Makefile option setting
# for Intel Fortran compiler
# FC = mpif90
# FFLAGS = -O3                # pure MPI processing
# FFLAGS = -O3 -openmp        # MPI/OpenMP processing

# for Intel Fortran compiler on MIC
# FC = mpif90
# FFLAGS = -O3 -mmic -align array64byte # pure MPI processing
# FFLAGS = -O3 -mmic -openmp -align array64byte # MPI/OpenMP processing

# for FX10 fujitsu compiler
FC = mpifrtpx
FFLAGS = -O3 -Kfast # pure MPI processing
# FFLAGS = -O3 -Kfast -Kopenmp # MPI/OpenMP hybrid

# for Gfortran
# FC = gfortran
# FFLAGS = -O3 -funroll-loops -msse3 -m64
```

Makefile.option
の中身

1. 並列数とモデルサイズを変更して計算時間の変化を計測せよ

- m_param.f90
 - モデルサイズ: 256*256*256 grid points
 - 使用するノード数: 8ノード(128コア)
 - Job文のprocとOMP_NUM_THREADSの値を変化させる
- Jobファイル

```
#!/bin/sh
```

ハイブリッド並列(8 pureMPI + 16 threads)の場合

```
#PJM -L "rscgrp=tutorial"
```

```
#PJM -L "node=8"
```

```
#PJM -L "elapsed=00:15:00"
```

```
#PJM --mpi "proc=8"
```

```
export OMP_NUM_THREADS=16
```

```
fapp -C -d prof -L 1 -lhwm -Hevent=Statistics mpiexec lpgparm -p 256MB -d 256MB -h 256MB -s 256MB -t 256MB  
./seism3d3n
```

- ./doc以下にあるマニュアルを参照
 - user guide
 - reference guide
- 古村孝志, 地震波伝播と強震動の大規模並列FDMシミュレーション, 東京大学情報基盤センタースーパーコンピューティングニュース, Vol11, pp.35-63, 2009.
- Mori F et al., Lecture Notes in Computer Science (LNCS 8969), pp.66-76, 2015.
- Mori F et al., 11th International Meeting High Performance Computing for Computational Science (VECPAR2014)
- Matsumoto M et al., Multi-scale Coupling Simulation of Seismic Waves and Building Vibrations Using ppOpen-HPC, Procedia Computer Science 51: 1514-1523 (2015)