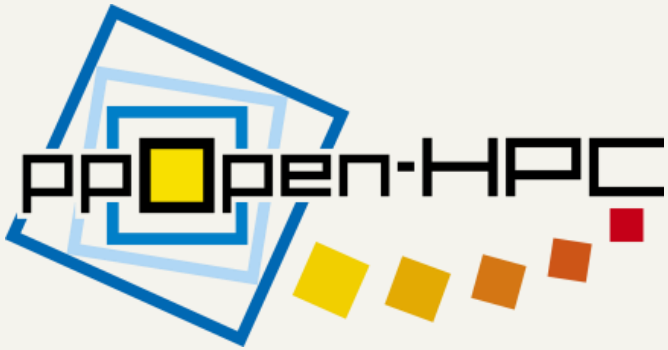



THE UNIVERSITY OF TOKYO



# 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

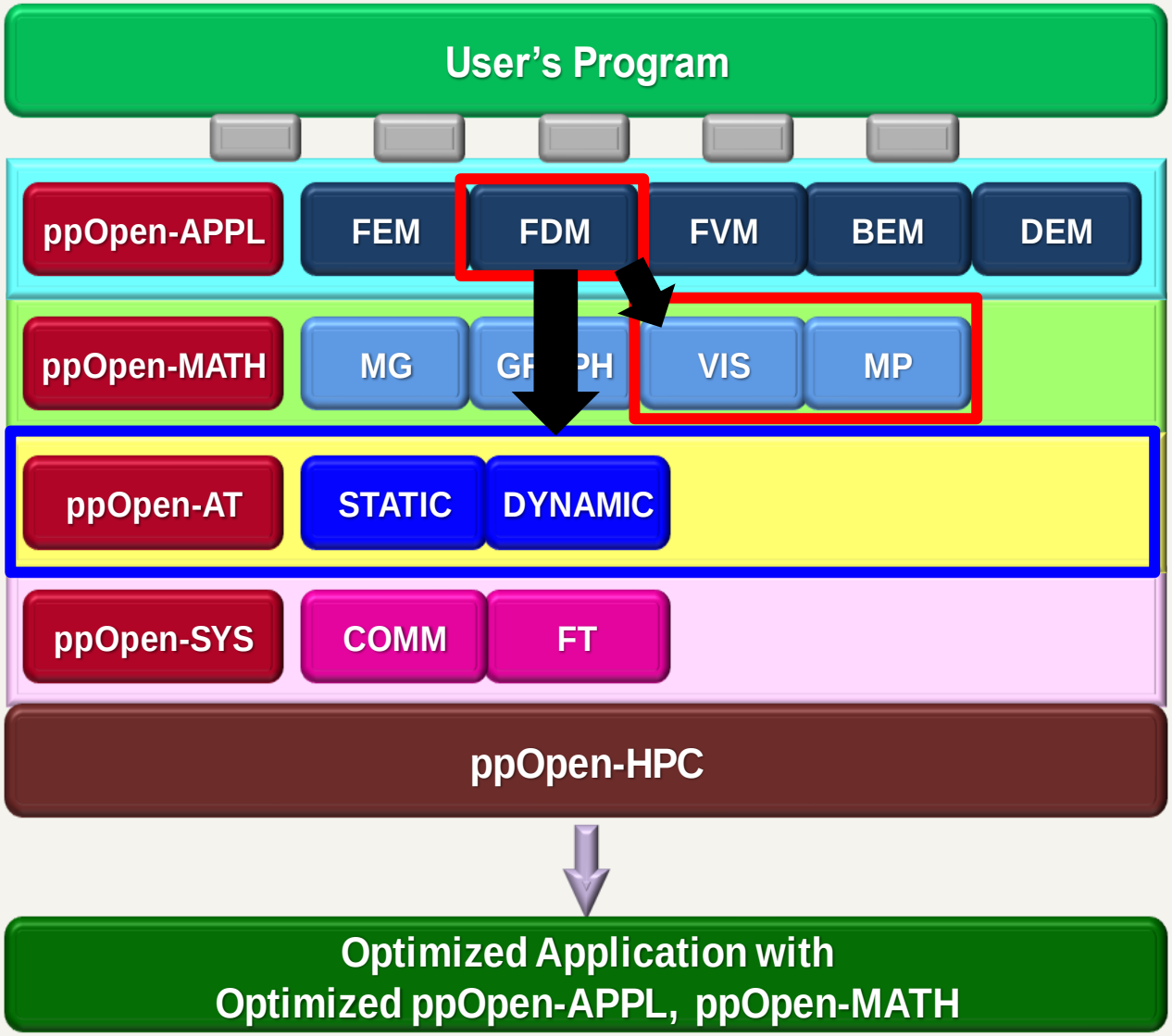
# 本日の目次

- はじめに
  - 概要
    - 弾性波動計算
    - ppOpen-APPL/FDM
  - 近年のスパコンのトレンド
    - 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

- ppOpen-APPL/FDMの演習
  - 利用方法
  - 演習

# 概要

# ppOpen-APPL/FDM library



# FDMによる弾性波動計算

- 構成方程式
  - 弾性体の釣り合いの式（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、$\sigma$: 応力、$\rho$: 密度、$f$: 外力

  - 等方完全弾性体の応力

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

λ, $\mu$ : Lame定数、$\delta$: クロネッカのデルタ

    - 歪みは変位の空間微分で求められる

$$e_{pq} = \frac{1}{2}\left(\frac{\partial u_p}{\partial q} + \frac{\partial u_q}{\partial p}\right), \quad (p, q = x, y, z) \quad (3)$$

$e$: 歪み

# 弾性波動の陽的計算

- 時間発展により波動伝播を進めるために、式(1)の速度変数を中間変数とする

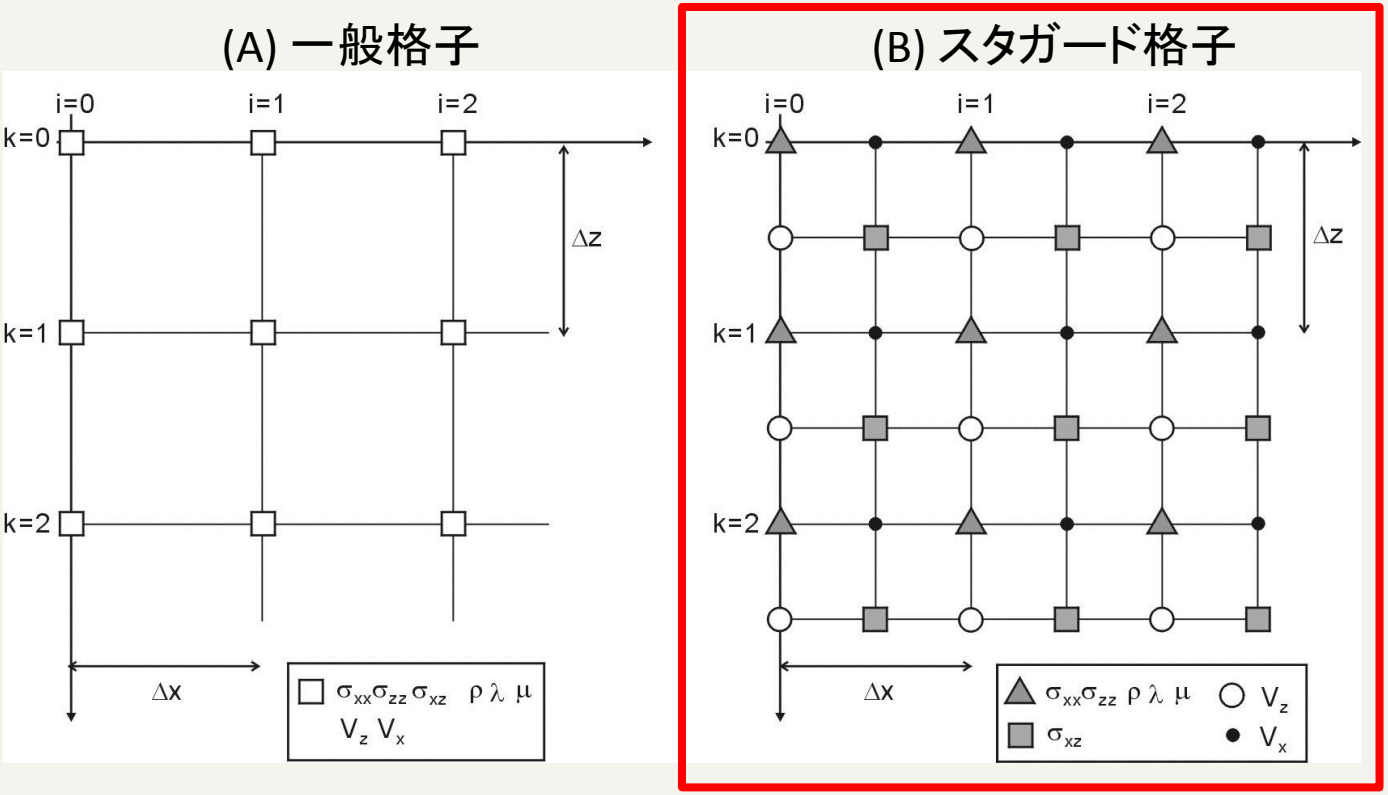
$$\dot{u}_p^{n+\frac{1}{2}} = \dot{u}_p^{n-\frac{1}{2}} + \frac{1}{\rho}\left( \frac{\partial \sigma_{xp}^n}{\partial x} + \frac{\partial \sigma_{yp}^n}{\partial y} + \frac{\partial \sigma_{zp}^n}{\partial z} + f_p^n \right) \Delta t, \ (p = x, y, z) \qquad (4)$$

$\dot{u}^{n+1/2}{}_p$ 粒子速度

- 式(2)と式(3)を結合した式(5)を用いて中央差分に基づく時間積分

$$\sigma_{pq}^{n+1} = \sigma_{pq}^n + \left[ \lambda \left( \frac{\partial \dot{u}_x^{n+\frac{1}{2}}}{\partial x} + \frac{\partial \dot{u}_y^{n+\frac{1}{2}}}{\partial y} + \frac{\partial \dot{u}_z^{n+\frac{1}{2}}}{\partial z} \right) \delta_{pq} + \mu \left( \frac{\partial \dot{u}_p^{n+\frac{1}{2}}}{\partial q} + \frac{\partial \dot{u}_q^{n+\frac{1}{2}}}{\partial p} \right) \right] \Delta t, \ (p,q) = (x,y,z) \qquad (5)$$

# 格子モデル




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

# FDMによる弾性波動計算

- FDMによる式(4)と(5)の空間微分
  - 中心差分による計算(2次精度、4次精度、8次精度)

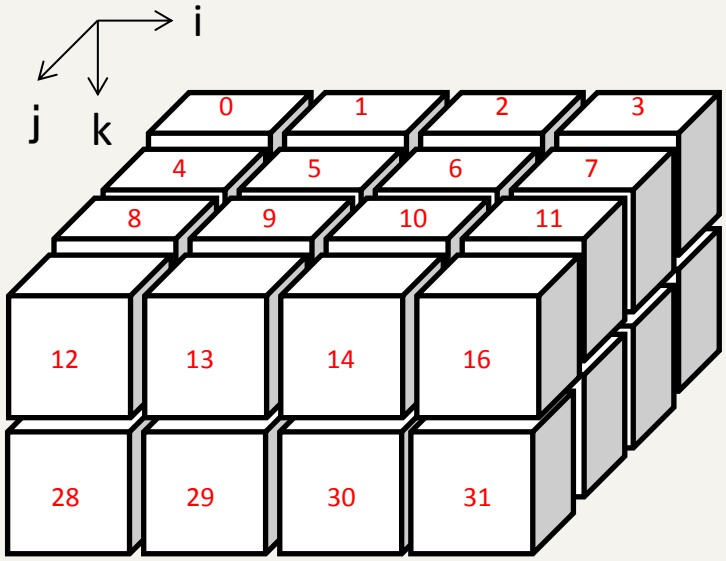
(2次精度)
$$\frac{d}{dx}\sigma_{pq}(x,y,z) \simeq \frac{1}{\Delta x}\left[\sigma_{pq}\left(x+\frac{\Delta x}{2},y,z\right)-\sigma_{pq}\left(x-\frac{\Delta x}{2},y,z\right)\right]$$

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

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

# ppOpen-APPL/FDM　概要

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
   - 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

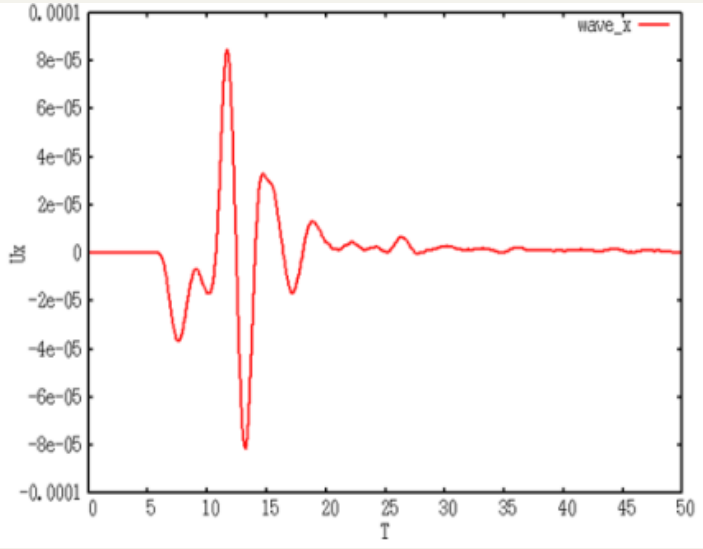



3次元領域分割
（赤文字：MPIランク）

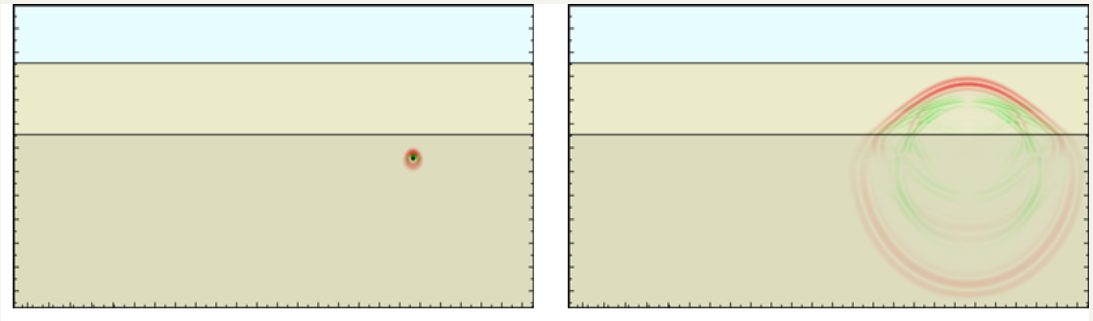
# ppOpen-APPL/FDM　概要 (cont.)

## 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



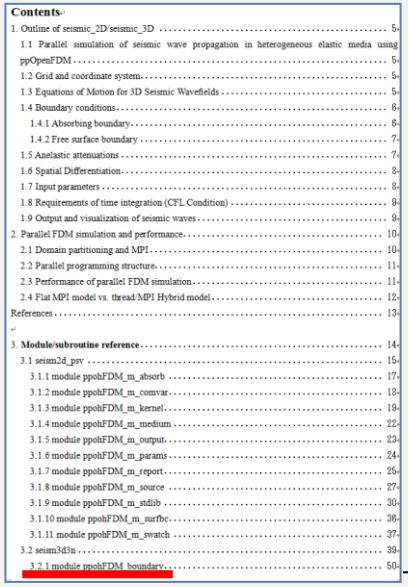

観測点の波形

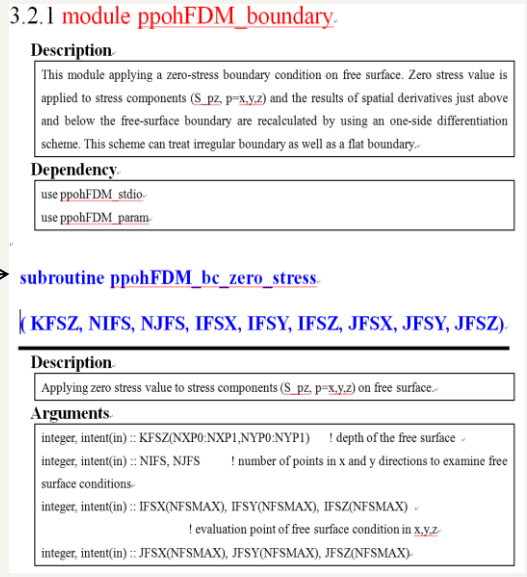


波動場のスナップショット

# ppOpen-APPL/FDM概要 (cont.)

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




- 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　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

 !-<< 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
```

モデルサイズ

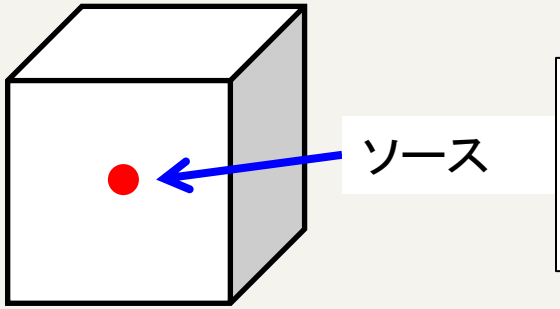
格子間隔

時間間隔

MPI領域分割

# 1. 入力パラメータの例 (ソースの設定、地下構造の設定)

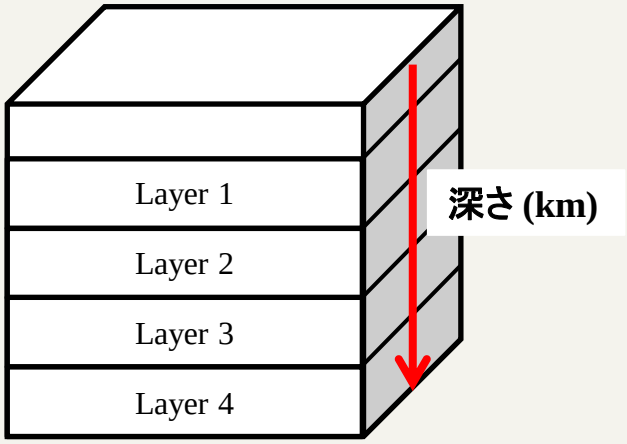
ソースの設定
（source.dat）




**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

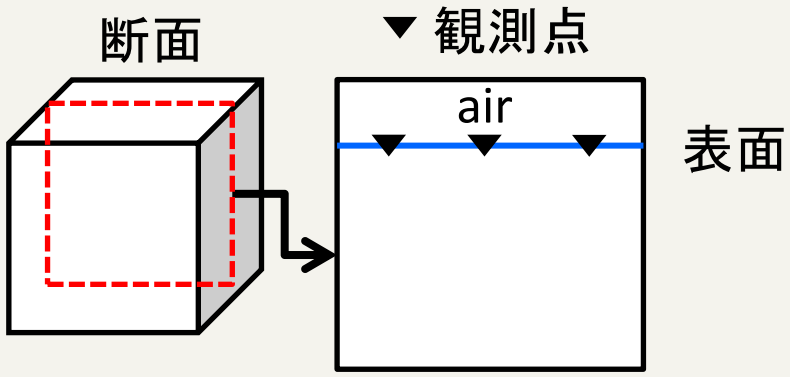



**medium.dat**

```
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
```

（注意）./examples/seismic_3D-exampleにこれら4つのパラメータファイルが用意されています。
./src/seismic_3D/1.ppohFDM-ppohVIS or 2.parallelにCOPY

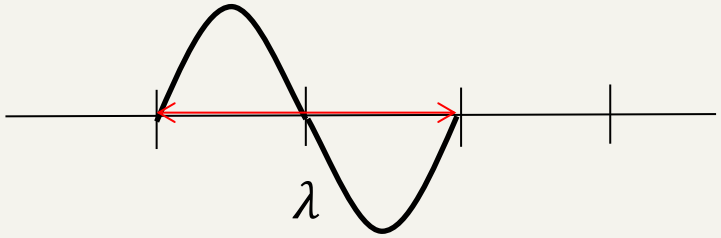
# 最大周波数と安定条件

## (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)

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




$$V_s^{min} = fm^{ax} \times \lambda^{min}$$

(伝わる速度)　(周波数)　(波長)

$$1.7 = fm^{ax} \times \mathbf{3.0}$$

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

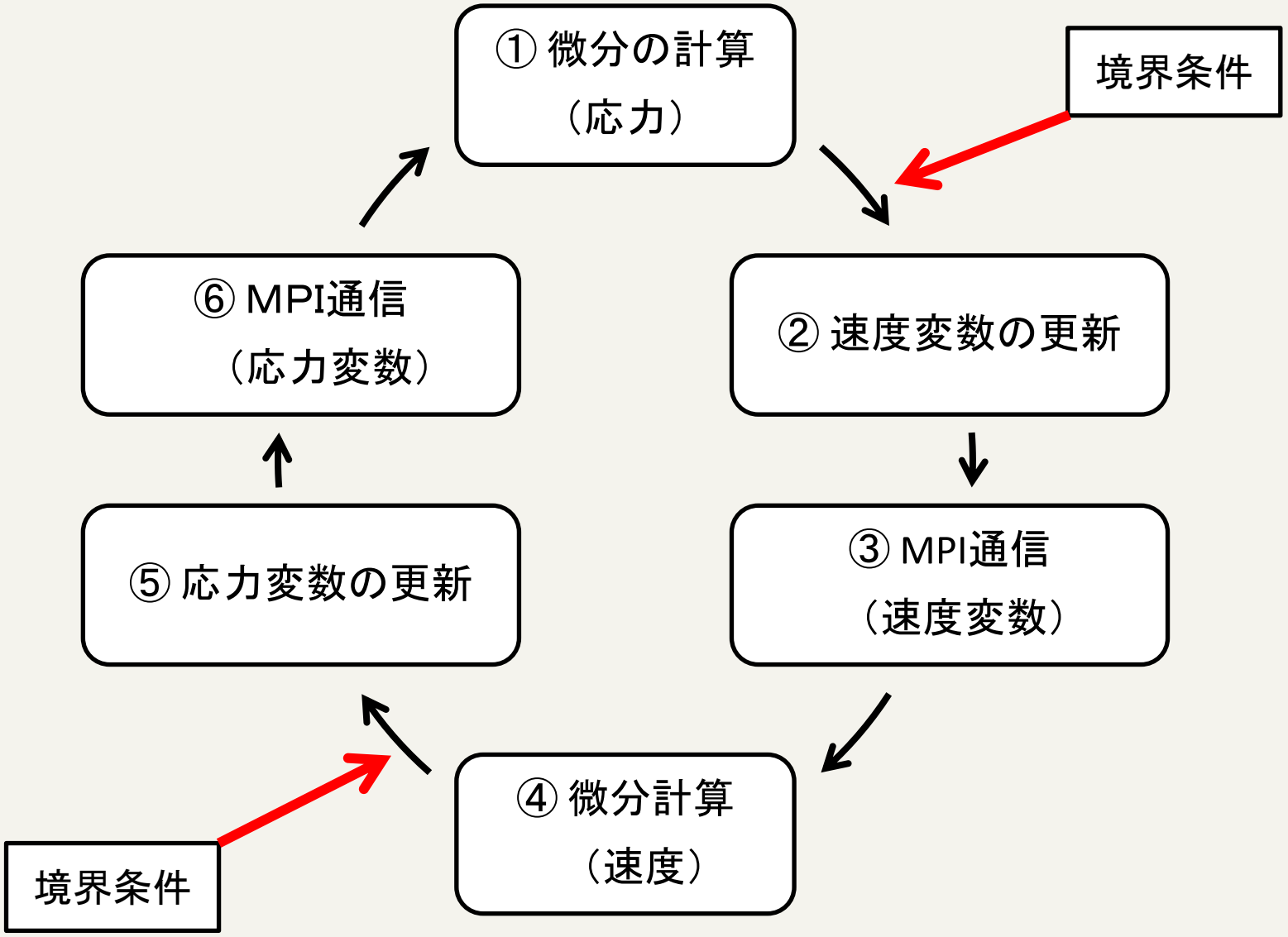
## (2) Δtの決め方

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

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

青: 未知数
黒: 既知数

# 2. 計算手順


① 微分の計算
（応力）
境界条件
② 速度変数の更新
③ MPI通信
（速度変数）
④ 微分計算
（速度）
境界条件
⑤ 応力変数の更新
⑥ MPI通信
（応力変数）

# 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
```

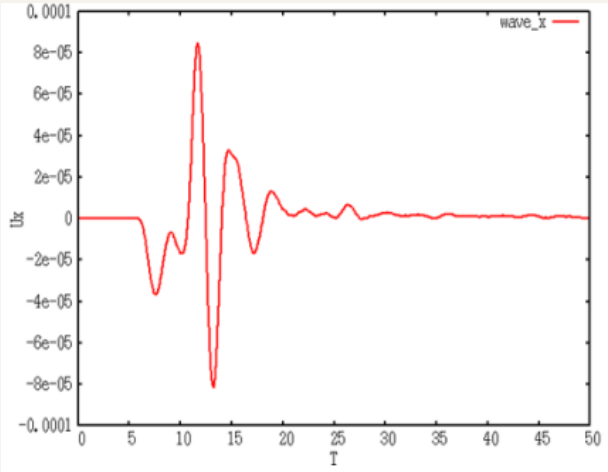
| | |
|---|---|
| SEISM3D3.prm | 計算パラメータ |
| SEISM3D3.WAV.*** | 観測点における波形 |
| SEISM3D3.SPS.*** | P波とS波の波動場 |
| SEISM3D3.SUR.*** | 表面上での波動場 |
| SEISM3D3.XY(XZ, YZ).*** | 各断面での波動場 |

# 3. 出力および可視化 (cont.)

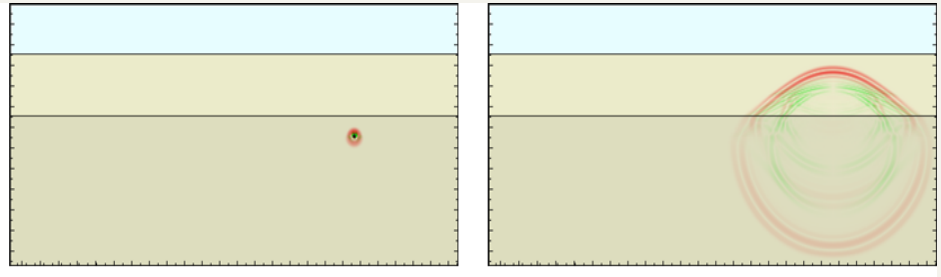
- ./tools/seismic_3D-toolsにおいてmakeすると4つ実行ファイル（catsnap, catwav, ppmxy3d3,rwav3d）が生成される

- % 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

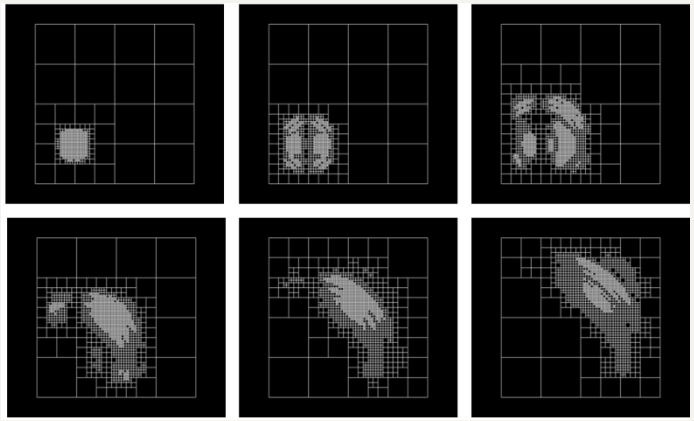


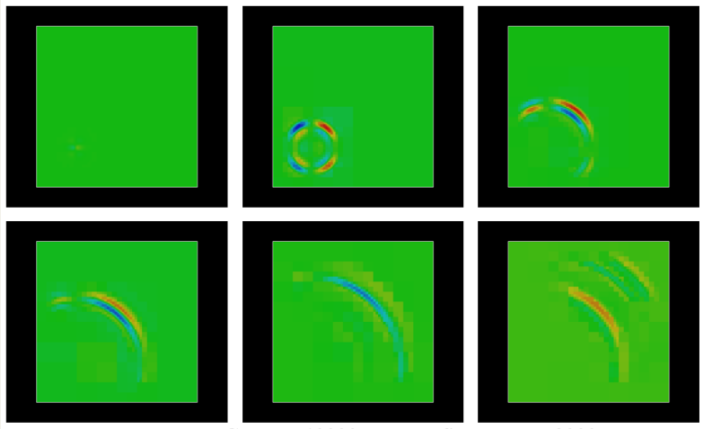
波動場のスナップショット
（P波：赤、S波: 緑）

# 可視化 ppOpen-MATH/VIS

- 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

- ppOpen-APPL/FDM 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かくなる ( ./examples/seismic_3D-example )
  - AVSやParaviewで可視化することができる





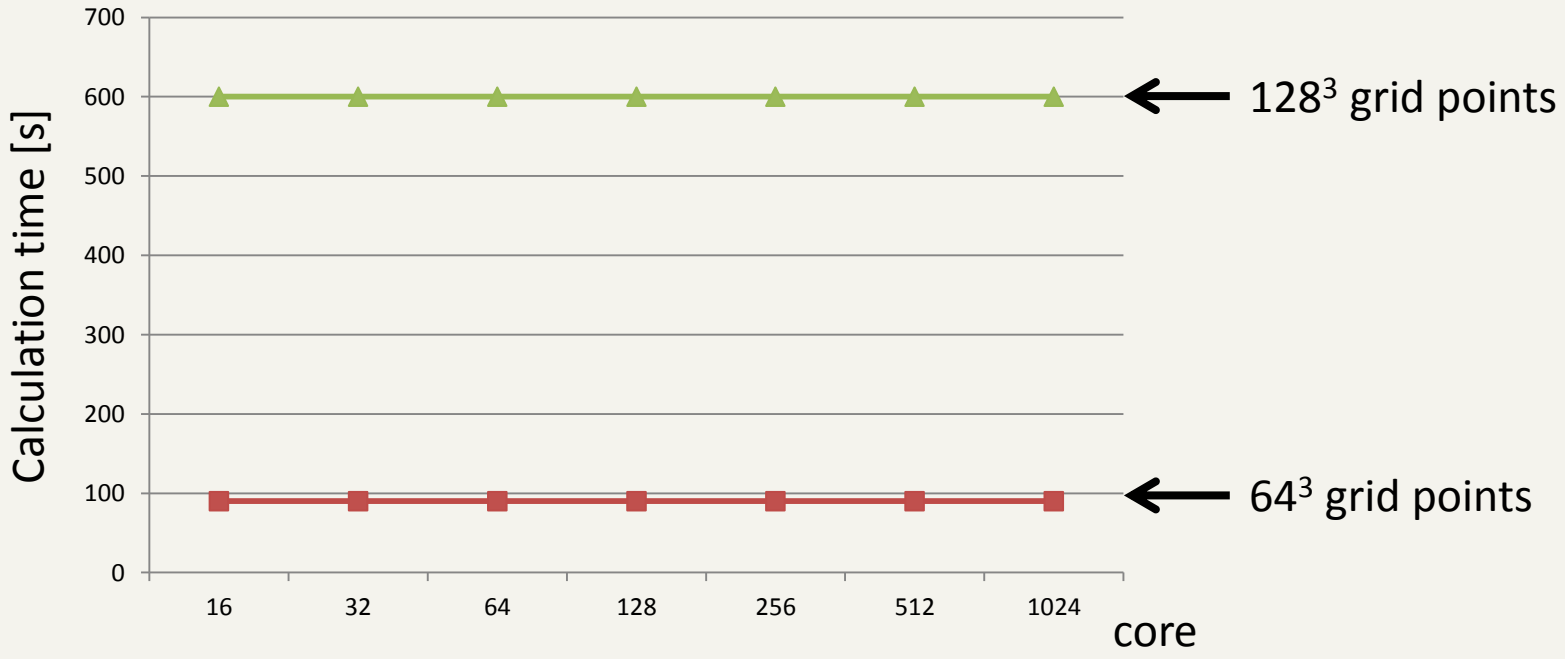
VXの速度場

# ヘ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

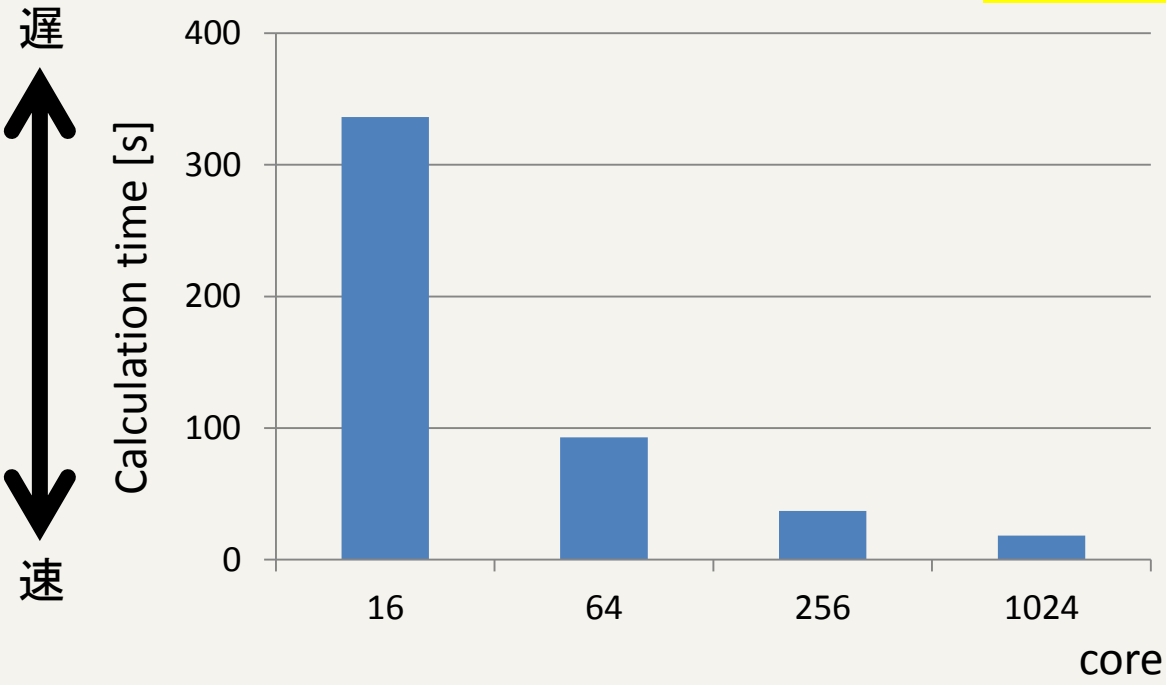
- モデルサイズ：$64^3$ , $128^3$ grid points
- 並列数：16～1024 コア
- 3D domain partitioning of MPI



# Strong scaling test in the flat 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

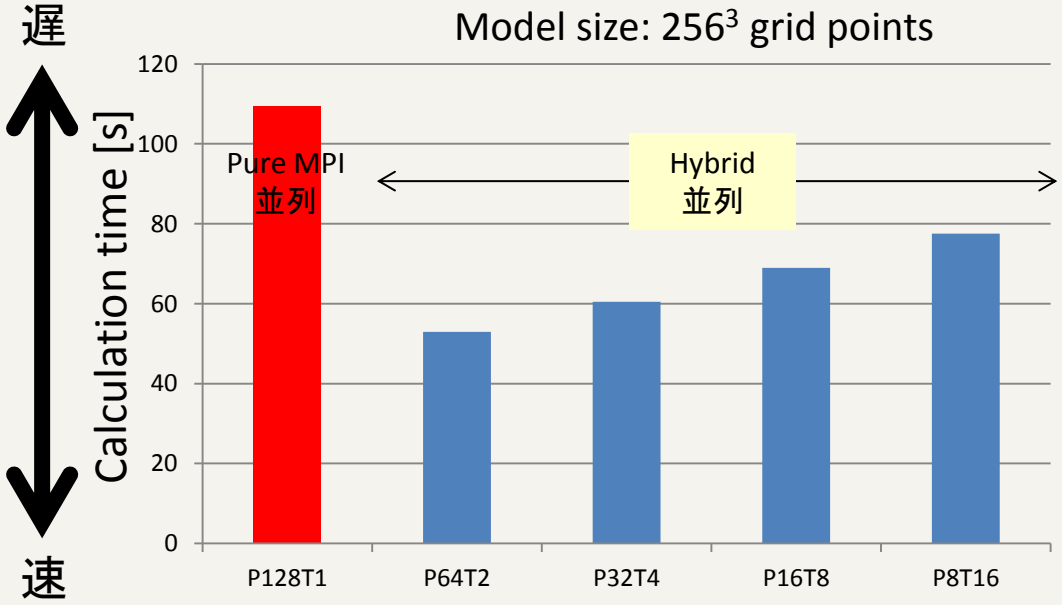
$256^3$ grid points



# 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
  - $256^3$ 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



# スパコンのトレンド

| 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, Intel Xeon Phi 31S1P 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, NVIDIA K20x 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 | Swiss National Supercomputing Centre (CSCS) Switzerland | **Piz Daint** - Cray XC30, Xeon E5-2670 8C 2.600GHz, Aries interconnect , NVIDIA K20x Cray Inc. | 115,984 | 6,271.0 | 7,788.9 | 2,325 |
| 7 | King Abdullah University of Science and Technology Saudi Arabia | **Shaheen II** - Cray XC40, Xeon E5-2698v3 16C 2.3GHz, Aries interconnect Cray Inc. | 196,608 | 5,537.0 | 7,235.2 | 2,834 |
| 8 | Texas Advanced Computing Center/Univ. of Texas United States | **Stampede** - PowerEdge C8220, Xeon E5-2680 8C 2.700GHz, Infiniband FDR, Intel Xeon Phi SE10P Dell | 462,462 | 5,168.1 | 8,520.1 | 4,510 |
| 9 | Forschungszentrum Juelich (FZJ) Germany | **JUQUEEN** - BlueGene/Q, Power BQC 16C 1.600GHz, Custom Interconnect IBM | 458,752 | 5,008.9 | 5,872.0 | 2,301 |
| 10 | DOE/NNSA/LLNL United States | **Vulcan** - BlueGene/Q, Power BQC 16C 1.600GHz, Custom Interconnect IBM | 393,216 | 4,293.3 | 5,033.2 | 1,972 |

Top500.rog/lists/2015/06/

**From Top 500 (2015.06)**
Many CoreマシンとMany Coreマシン＋Accelerator / Co-processor を搭載しているマシンの変動

- Top 500はTop7以外は変わらない
- 75システム（2014.11）から90システム(2015.06)に増加
  - 内4システムが両方を搭載

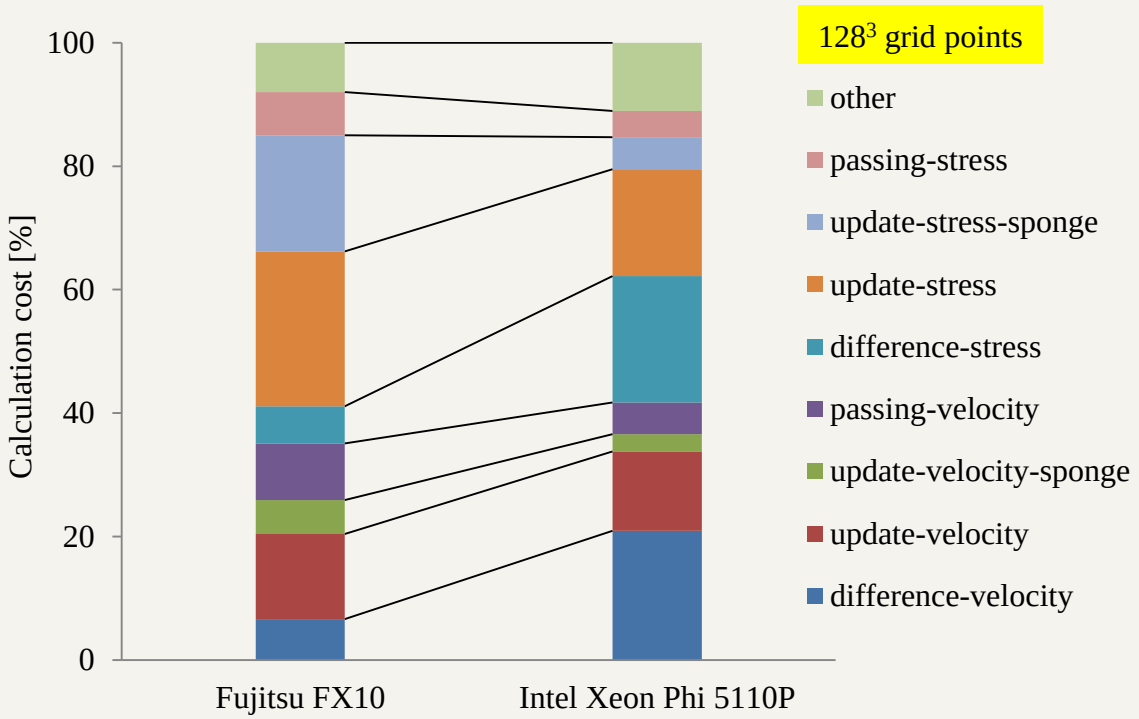
ちなみに、
Post T2K@東大・筑波大は
Intel Phi Xeon Co-processorを搭載

# 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 |

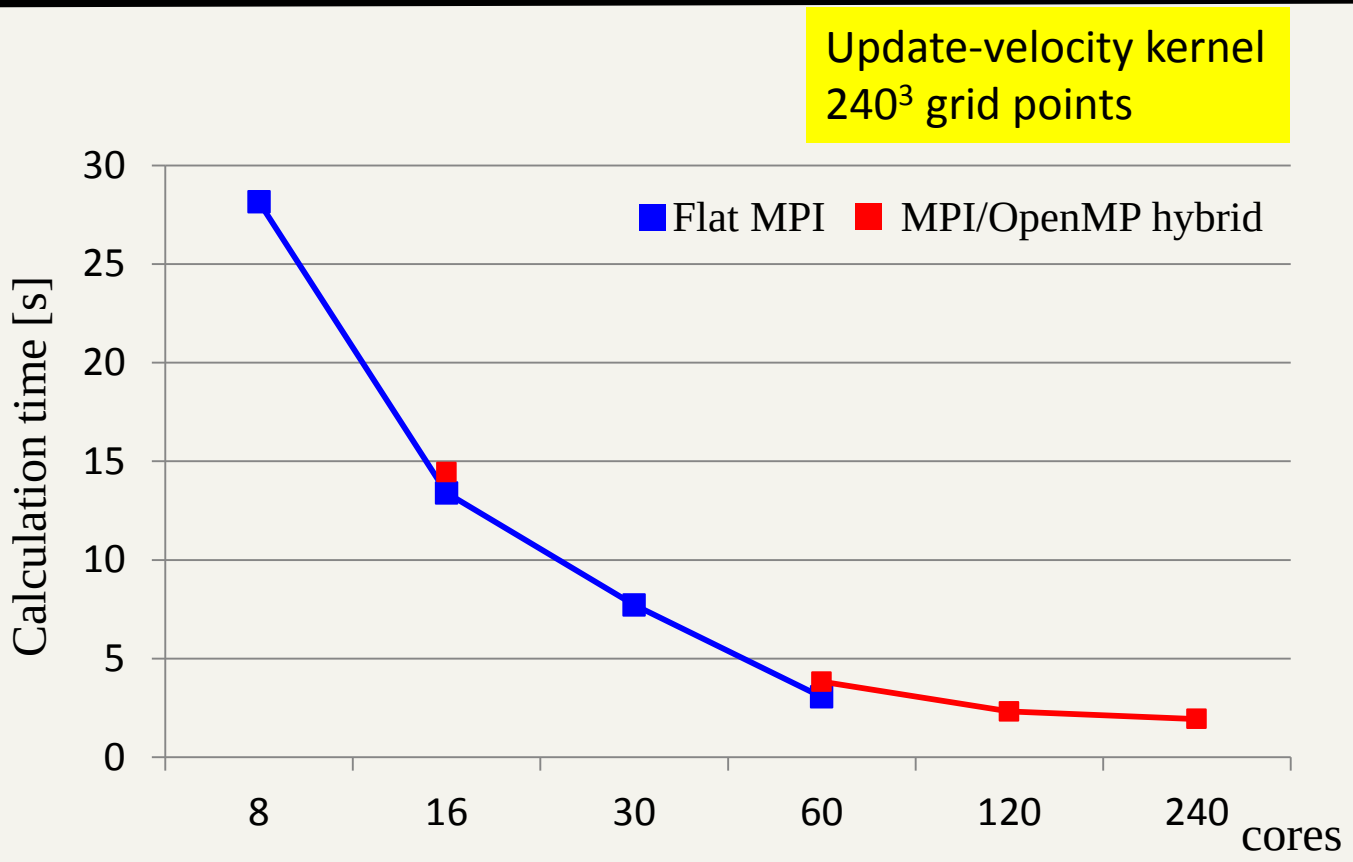
# Calculation costs

- 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 $128^3$ grid points**

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

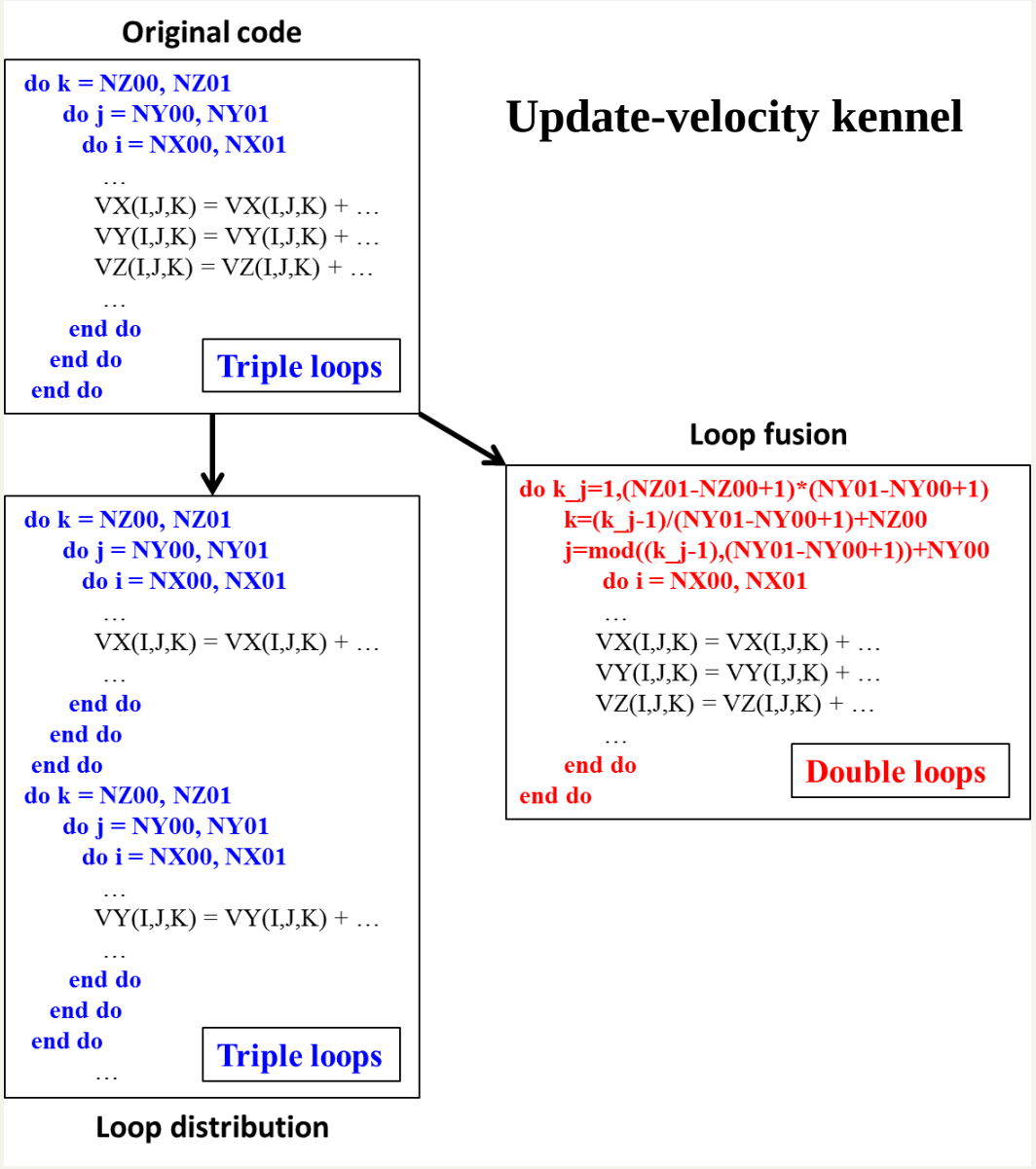



Xeon Phi のNative modeで実行

- 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

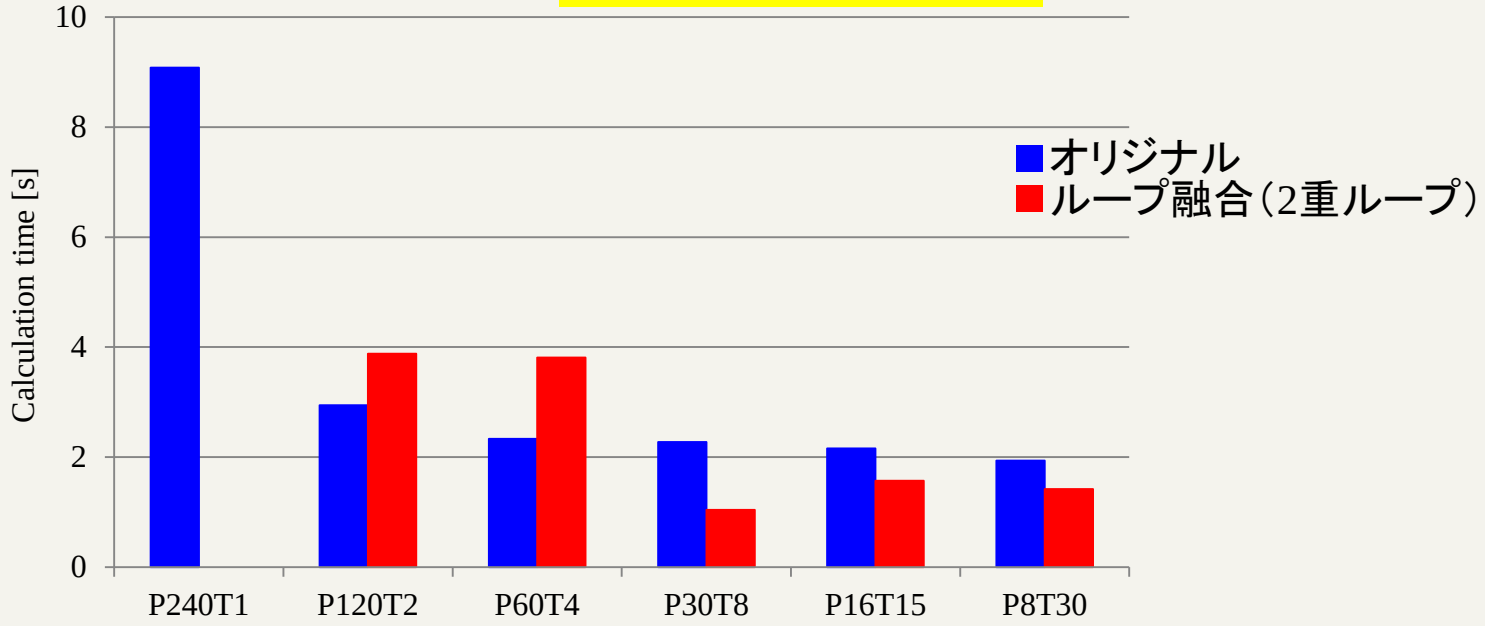
- ループの最適化
  - ループ融合 (Loop fusion)
  - ループ分割 (Loop distribution)



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

Update-velocity kernel
$240^3$ grid points

P: MPI process
T: OpenMP thread



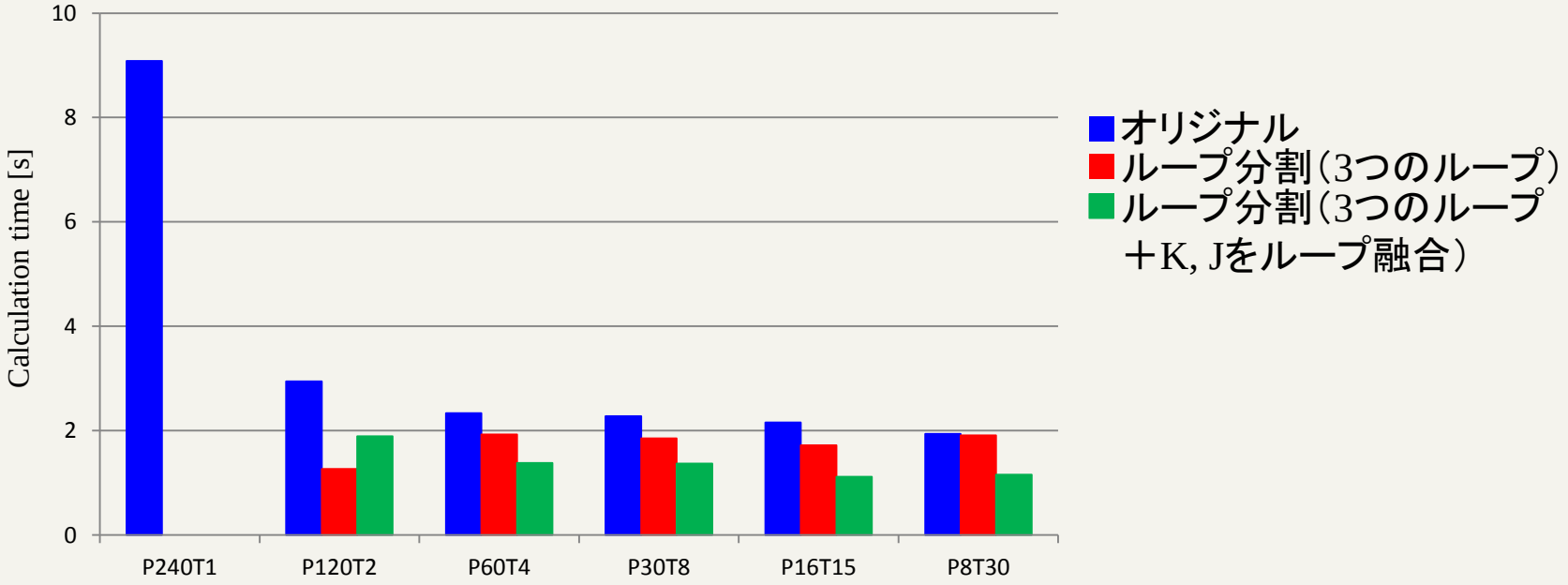

ループ融合によるパフォーマンス（240コアを使用した場合）

- 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
$240^3$ grid points

P: MPI process
T: OpenMP thread




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 **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）** の方が高速化
  - 1.9倍高速化（P16T15)
- Update-stress kernelも同様である

# B/F値の低減

- 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性能を得ることが難しい

  → 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                          Calculate Derivatives
  do I=1,NX                              kernel
   …
   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
```

```
do K=Nz00,Nz01
 do J=Ny00,Ny01
  do I=Nx00,Nx01                          Update-velocity
   …                                           kernel
   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)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  ,J,K))*C40/dx-(Sxx(I+2,J,K)-Sxx(I-1,J,K))*C41/dx
   DxSxy0=(Syy(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
```

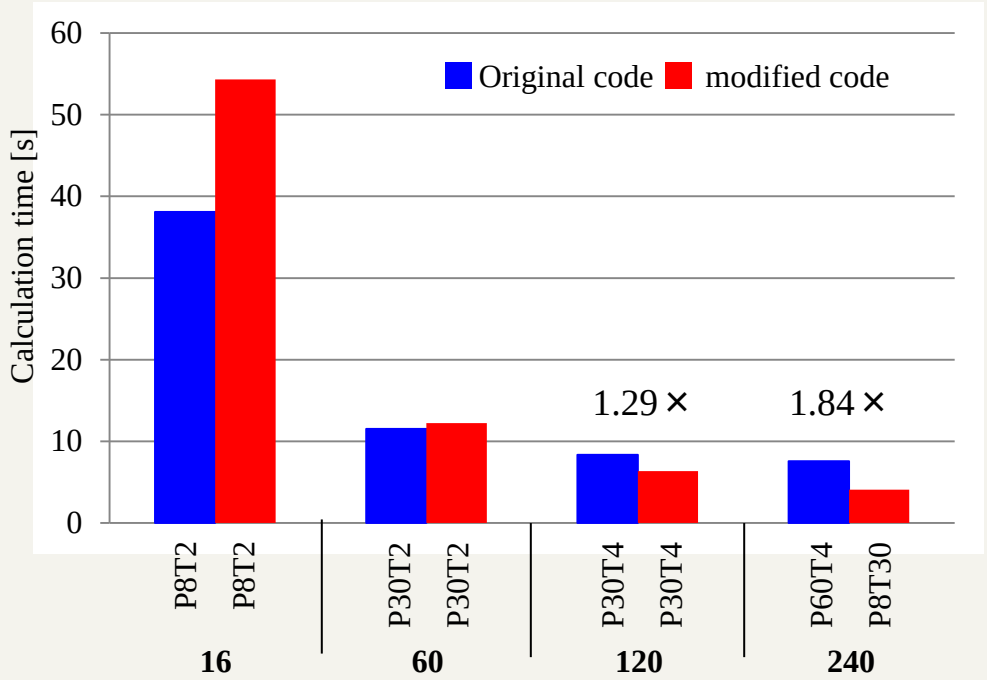
(update-velocity)要求**B/F=0.4,** (update-stress)要求**B/F=0.4**

# Original code and Modified code
## MPI/OpenMP hybrid parallel computing

**Xeon Phi 5110P @ERI**

Update-velocity kernel
$240^3$ grid points

P: MPI process numbers
T: OpenMP thread numbers



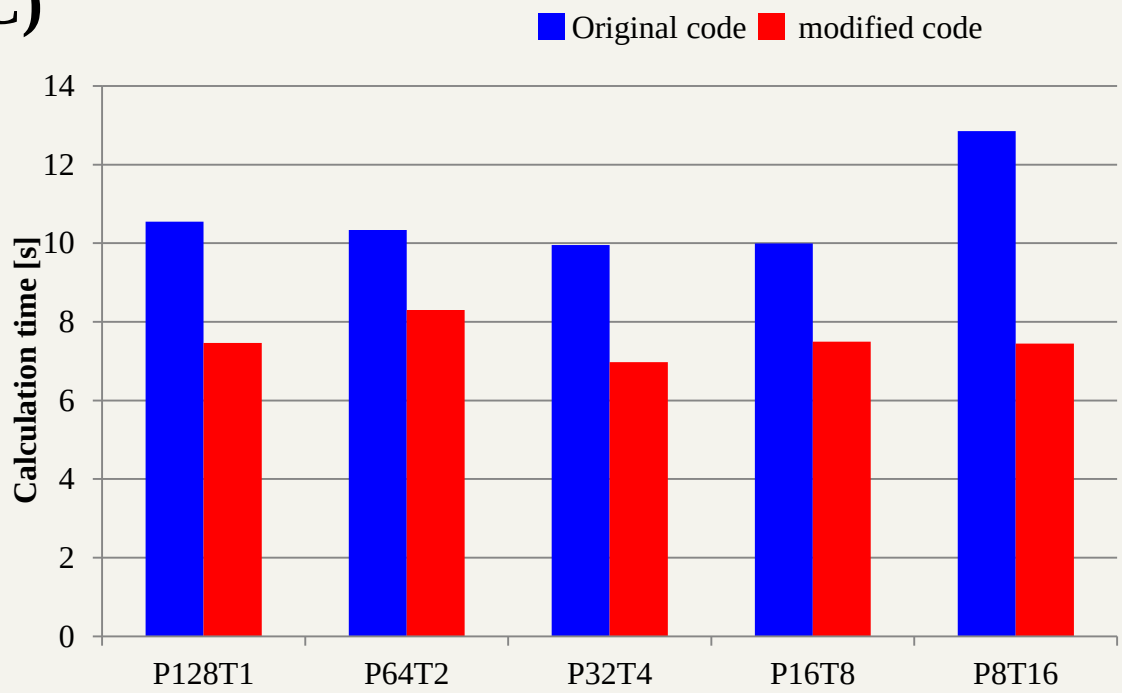

- MPI/OpenMP Hybrid 並列において、Modified codeがOriginal codeより高速になるのはHyper Threadingを使用したときであった
  - 1.9倍高速化（P8T30)
- Update-stress kernelにおいても同傾向

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

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

Update-velocity kernel
$512^3$ grid points

P: MPI process
T: OpenMP thread




FX10におけるOriginal code と Modified codeの比較

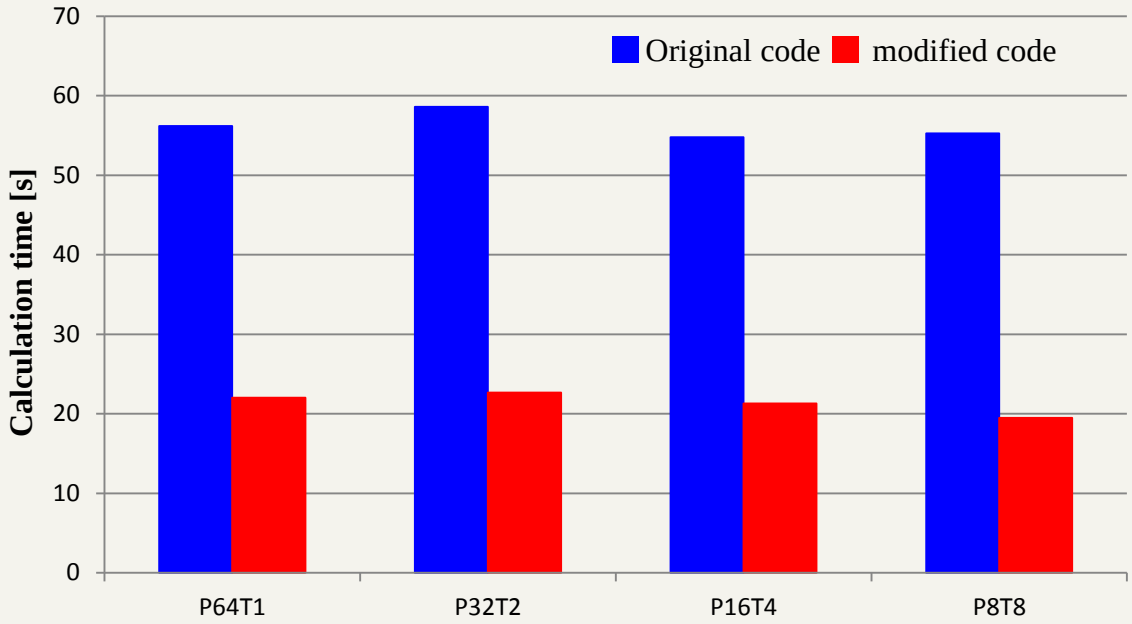
- Original codeの最速であるP32T4においてModified codeは更に1.4倍高速化される
- Update-stress kernelにおいても同傾向

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

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

Update-velocity kernel
$256^3$ 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倍高速化される
- Update-stress kernelにおいても同傾向

# まとめ

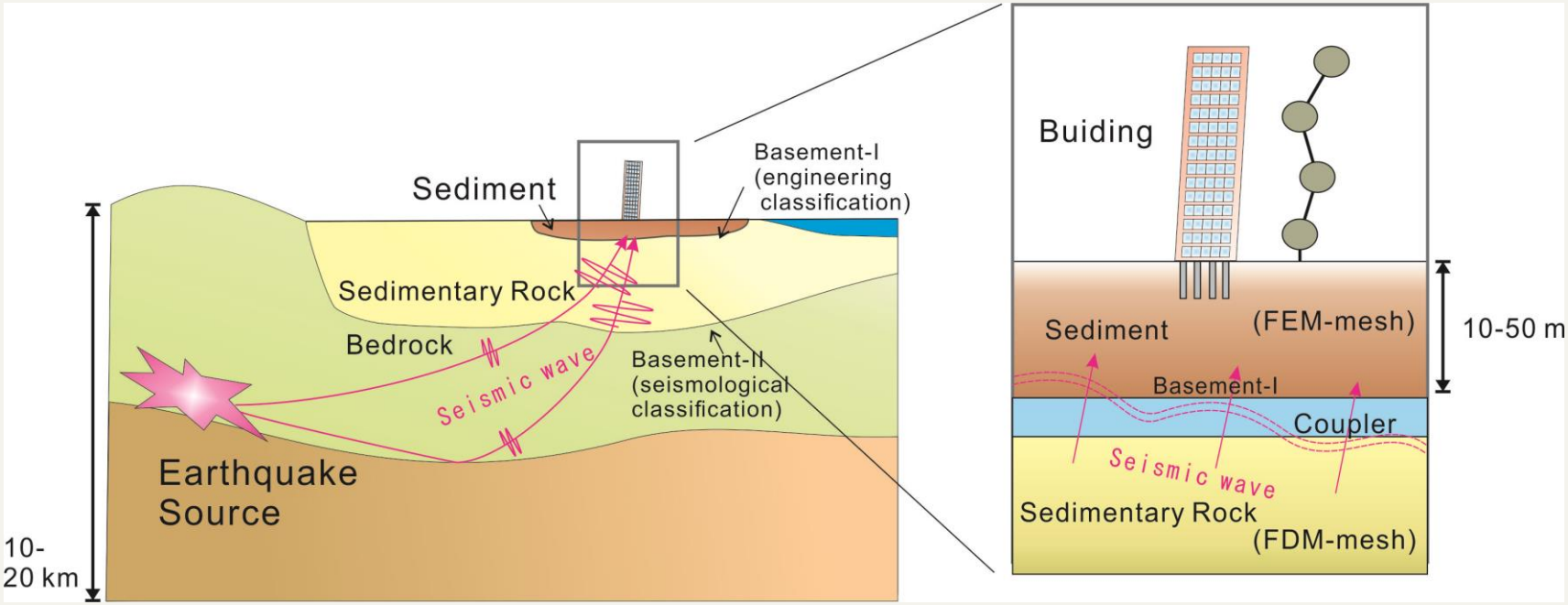
- Pure MPI並列 vs Hybrid並列
  - Hybrid並列の方が有効的である
- Intel Xeon Phi Coprocessorの最適化
  - チューニングとしてループ分割が全体のパフォーマンスを考えたとき有効的である
  - スレッド数の増加に対して、ループ分割＋ループ融合が安定的に効果が見込める
- B/F値の低減
  - ヘテロな環境においてメモリ律速から緩和させることによってCPU性能が顕在化してきた可能性がある
  - モデルサイズの違いがあることが推測される

# 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

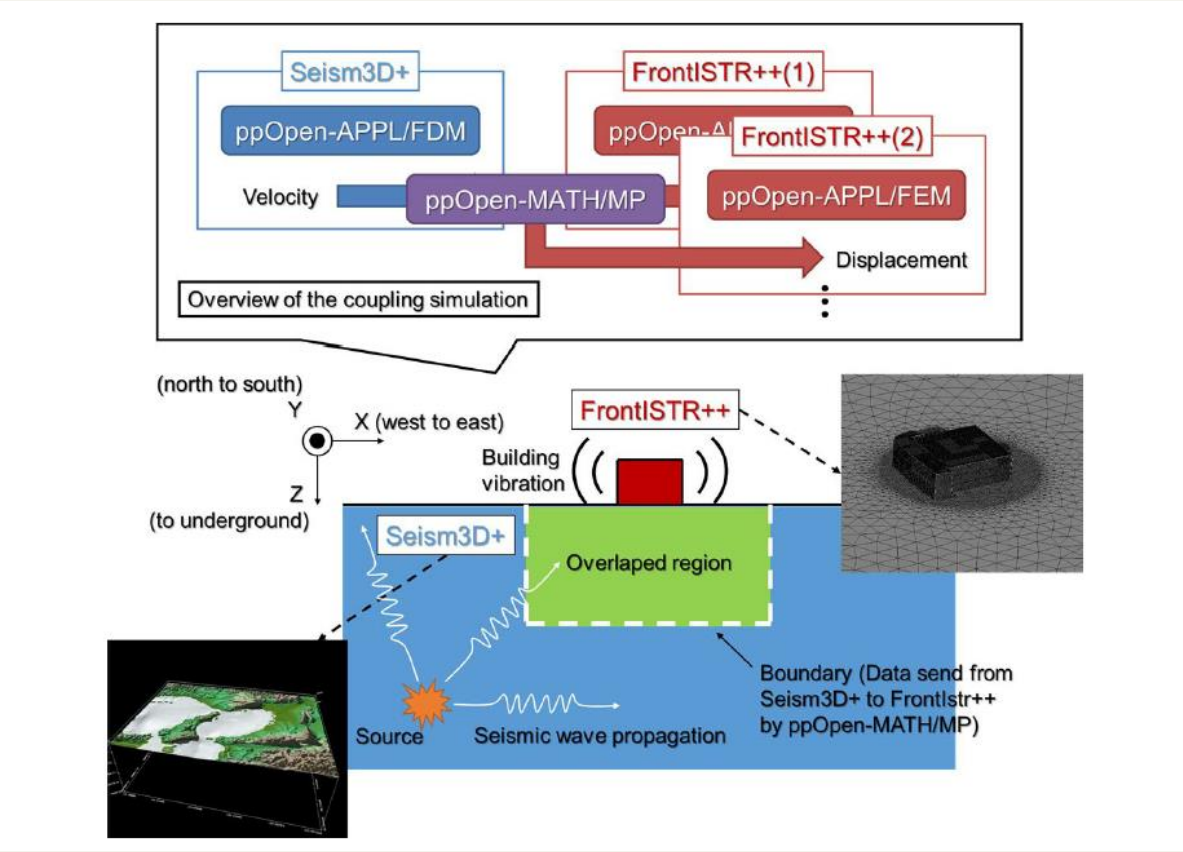
- ppOpen-HPCプロジェクト内で開発されているppOpen-MATH/MP　couplerを用いて差分法（地震動→ ppOpen-APPL/FDM）と有限要素法（構造解析→ ppOpen-APPL/FEM）を連成し、大規模連成計算をおこなっている




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

# Overview of Coupling simulation

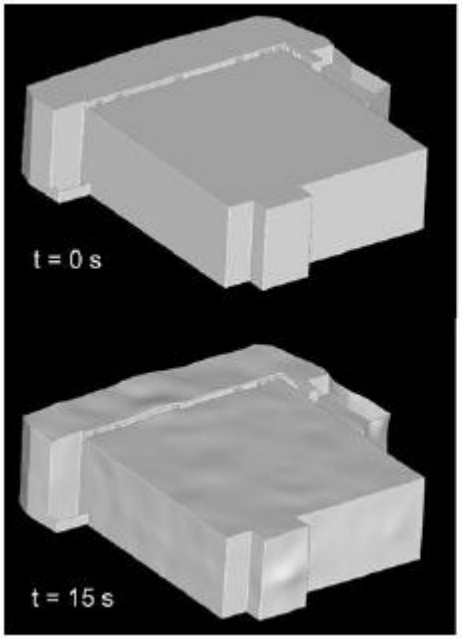
- **淡路**を震源とした地震を想定し、ポートアイランドにある**計算科学研究機構　京コンピュータの建屋**への影響をシミュレーションをおこなった

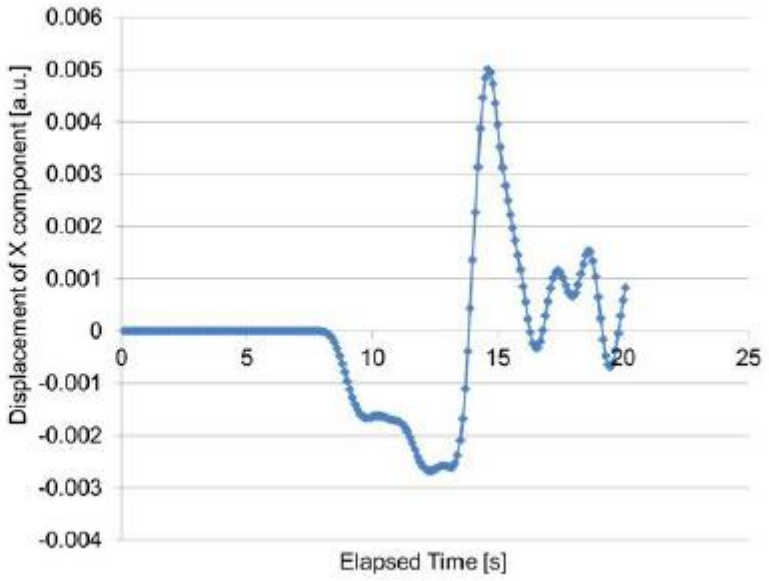


# 大規模連成解析の解析例

## 地震動


## 構造解析



# ppOpen-APPL/FDM
# 利用方法と演習

# はじめに

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

＊以下のファイルがインストールされていることを必須としています
Putty (in windows)
WinSCP (in windows)
FUJITSU Software Development
Paraview

**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 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

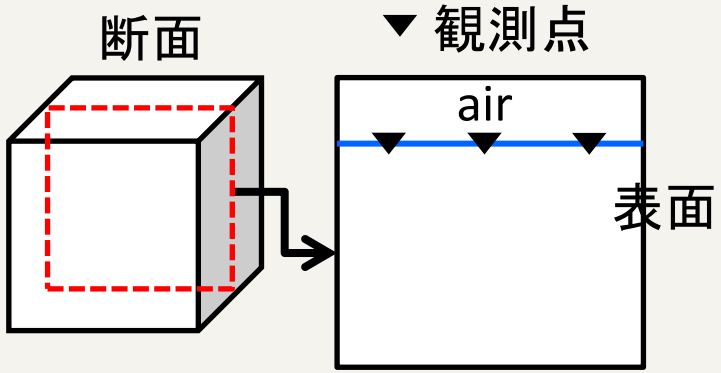
並列コード

# 入力パラメータ設定(1)

- 3つの入力パラメータファイル

./examples/seismic_3D-example
にあるのでコピー

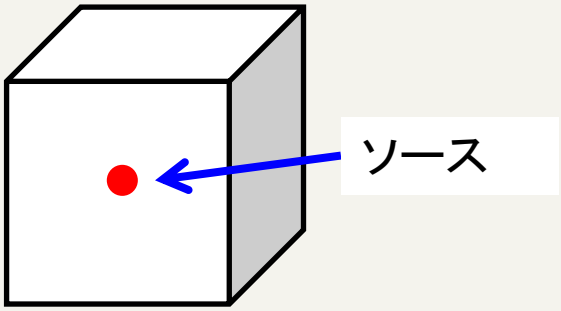
1. 観測点(X, Y, Z (km))の設定




**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**

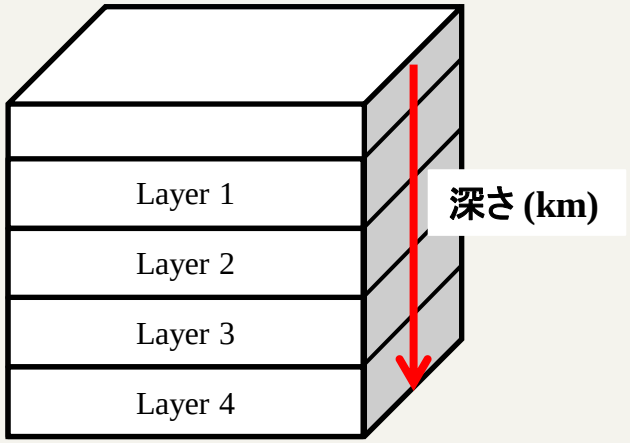
```
16.0 16.0 16.0      # ソースの位置: X(km), Y(km), Z(km)
0.0 45.0 90.0       # 断層パラメータ
1.0 2.0             # ソース時間: at, t0 (s)
```

# 入力パラメータ設定(2)

3. 地下構造モデル（深さ(depth), 密度(RO), 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
```




./examples/seismic_3D-example
にあるのでコピー

# 計算パラメータ設定

## 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

./examples/seismic_3D-example にあるのでコピー

```
!-- << 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

 !-<< 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
```

モデルサイズ

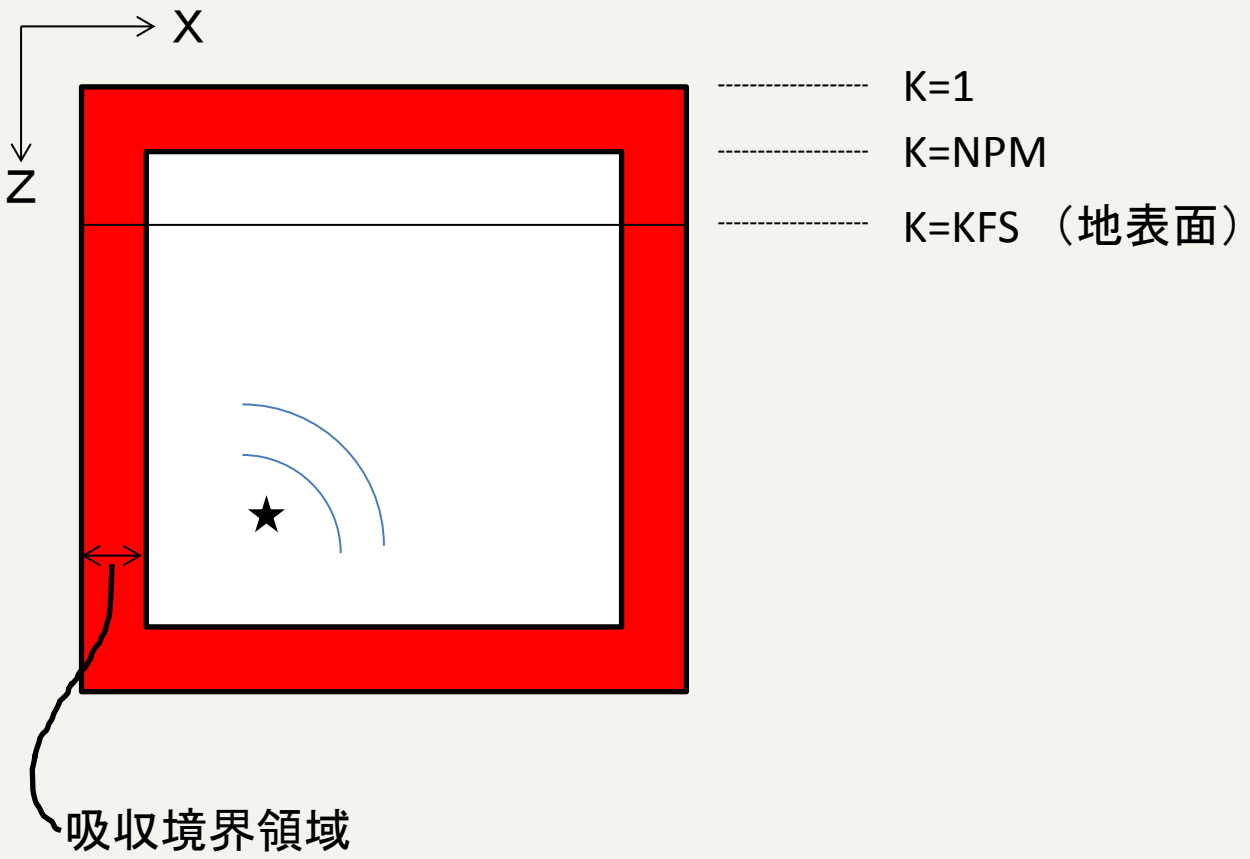
タイムステップ

格子間隔

時間間隔

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。

# 演習(1)

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 = 2.0
MaxVoxelCount = 10000
MaxRefineLevel = 20
[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

(注意）
Makefile.inにMATH/VISライブラリのコンパイル先が書かれている。各ユーザはディレクトリを場所を絶対パスで書く

- 実行
  - % cd ./src/seimic_3D/1.ppohFDM-ppohVIS
  - %pjsub job

```
#!/bin/sh

#PJM -L "rscgrp=**"
#PJM -L "node=**"
#PJM -L "elapse=**:**:**"
#PJM -g **
#PJM --mpi "proc=**"

mpiexec ./seism3d3n
```

**はユーザがパラメータに応じて変更

./etc/jobがあるのでコピー

**m_param.f90で設定したNPとproc数を同じにする**

# 演習(1)

- 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

- ローカルにデータを移動させ、Paraviewを使って可視化する

# 演習(2) pureMPI並列

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,・・・
   - 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に書き込む

```
!! Velocity Update
call fapp_start("region2",1,1)
call ppohFDM_update_vel      ( 1, NXP, 1, NYP, 1, NZP )
call fapp_stop("region2",1,1)
```

# 演習（2） pureMPI並列

- Job ファイル

```
#!/bin/sh

#PJM -L "rscgrp=**"
#PJM -L "node=**"
#PJM -L "elapse=**:**:**"
#PJM -g **
#PJM --mpi "proc=**"

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

**はユーザがパラメータに応じて変更

./etc/jobがあるので適宜修正してください

- プロファイルを保存するディレクトリが必要
  - ./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する

# 演習（3）Hybrid parallel computing

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） or 4ノード（64コア）に固定
       - IP, JP, KPの値を変化させる(プロセス数）
       - Job文のexport OMP_NUM_THREADSの値を変化させる
   - Jobファイル

```
#!/bin/sh

#PJM -L "rscgrp=***"
#PJM -L "node=**"
#PJM -L "elapse=**:**:**"
#PJM -g **
#PJM --mpi "proc=**"

export OMP_NUM_THREADS=**

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

**はユーザがパラメータに応じて変更

./etc/jobがあるので適宜修正してください

# 参照文献

- ./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)