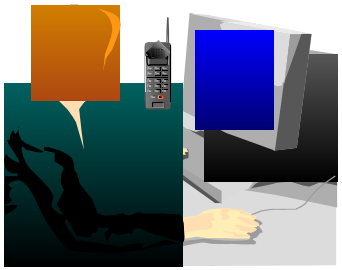
並列型パーソナルコンピュータによるFDTD計算

MPI: Message Passing Interface

名古屋工業大学
王　建青

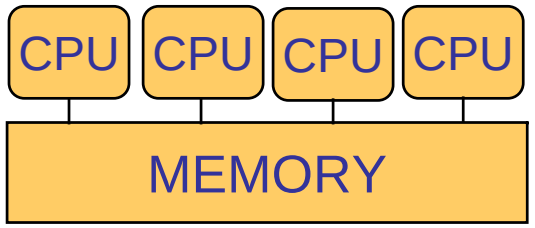


背景

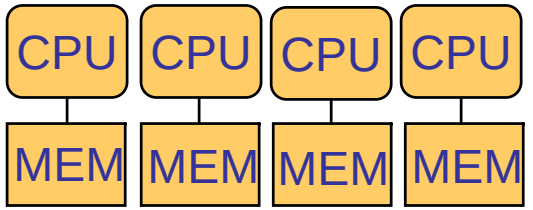
- FDTD法の特徴：領域分割型解法 ➡ 並列化

- 並列化の種類

  複数のCPUが単一のメモリを
  共有する共有メモリ型



  CPUごとにメモリをもたせる
  分散メモリ型

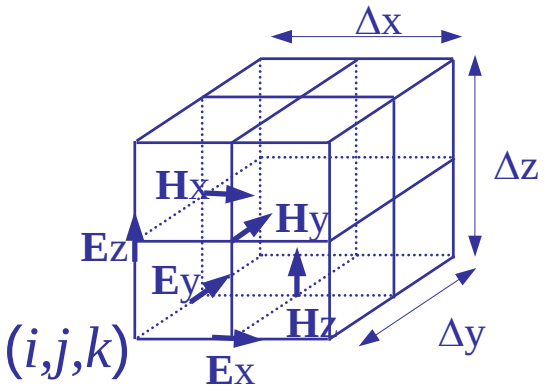


- パーソナルコンピュータの廉価化
  FDTD法の特徴
  ⟶ 分散メモリ型が有効

内容

- FDTDアルゴリズムの並列化の原理を述べる
- パーソナルコンピュータを用いた
  ８並列の構築例を紹介する

FDTDアルゴリズムの特徴



$$E_z^{n+1}(i,j,k)=\frac{1-\frac{\sigma\Delta t}{2\varepsilon}}{1+\frac{\sigma\Delta t}{2\varepsilon}}E_z^{n}(i,j,k)$$

$$+\frac{\frac{\Delta t}{\varepsilon}}{1+\frac{\sigma\Delta t}{2\varepsilon}}\left[\frac{H_y^{n+1/2}(i,j,k)-H_y^{n+1/2}(i-1,j,k)}{\Delta x}-\frac{H_x^{n+1/2}(i,j,k)-H_x^{n+1/2}(i,j-1,k)}{\Delta y}\right]$$

$$H_z^{n+1/2}(i,j,k)=H_z^{n-1/2}(i,j,k)$$

$$+\frac{\Delta t}{\mu}\left[\frac{E_y^{n}(i+1,j,k)-E_y^{n}(i,j,k)}{\Delta x}-\frac{E_x^{n}(i,j+1,k)-E_x^{n}(i,j,k)}{\Delta y}\right]$$

$$E^{n+1}(i,j,k)\Leftarrow H^{n+1/2}(i-1,j-1,k-1)$$

$$H^{n+1/2}(i,j,k)\Leftarrow E^{n}(i+1,j+1,k+1)$$

隣接の界成分だけ必要

ＦＤＴＤ計算アルゴリズム

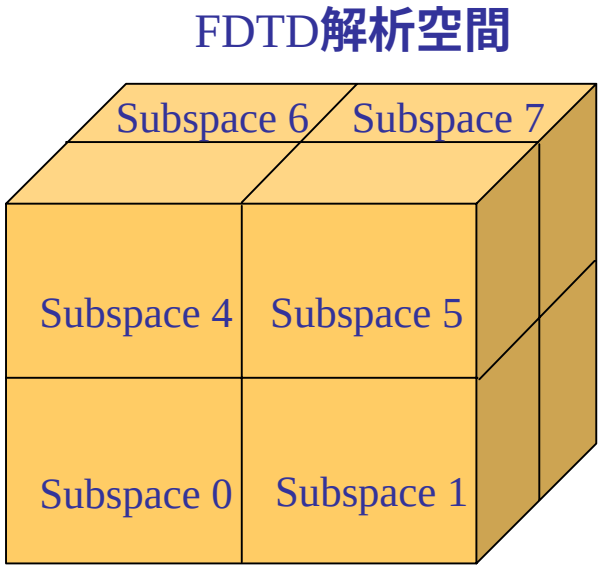
```
Program
Call INIT
Call SETUP
……
T=0
Do N=1, NSTOP
  Call E_FIELD
  Call E-PML
  T=T+DT/2
  Call H_FIELD
  Call H-PML
  T=T+DT/2
 End do
```

MPIを用いたFDTD並列計算

```
Program
Include ‘mpif.h’
……
Call MPI_INIT

T=0
Do N=1,NSTOP
  Call E
  Call E-PML
  Call MPI_SEND (E)
  Call MPI_RECV (E)
   T=T+DT/2

  Call H
  Call H-PML
  Call MPI_SEND (H)
  Call MPI_RECV (H)
  T=T+DT/2

End do

Call MPI_FINALIZE
```

MPI: Message Passing Interface

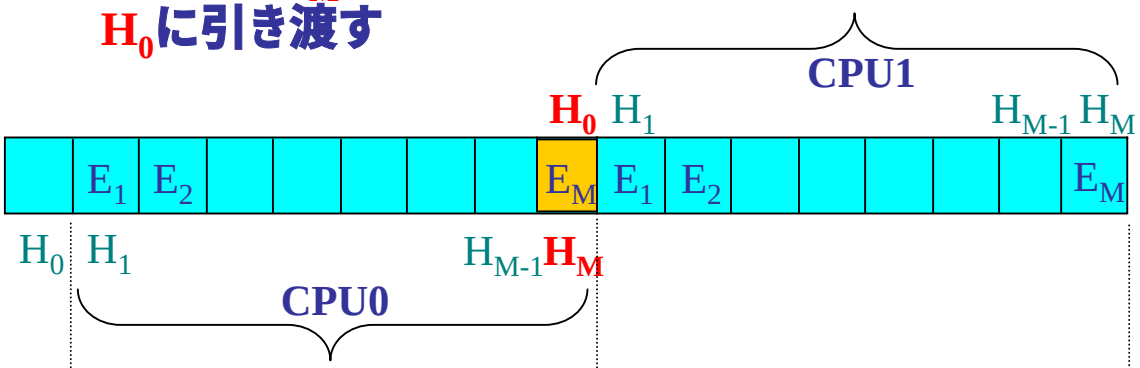
CPU0 → Subspace 0
CPU1 → Subspace 1
:
:
CPU7 → Subspace 7



サブ空間境界面でのデータ送受信　一次元の場合

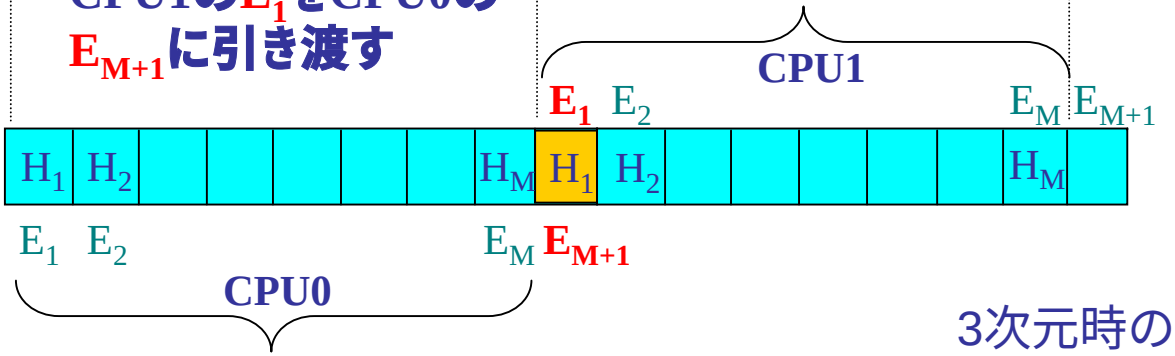
E-Field Computation : $E^{n+1}(i,j,k) \Leftarrow H^{n+1/2}\left(i-1,j-1,k-1\right)$

CPU0のH_MをCPU1のH_0に引き渡す



H-Field Computation : $H^{n+1/2}(i,j,k) \Leftarrow E^{n}\left(i+1,j+1,k+1\right)$

CPU1のE_1をCPU0のE_{M+1}に引き渡す

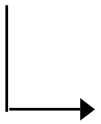


3次元時の定義：
E(II+1,JJ+1,KK+1,3)
H(0:II, 0:JJ, 0:KK,3)

プログラミング技法①

c　MPI 送受信のための準備

```
c　プロセッサID
      integer myrank
c　エラーコード
      integer ierr

c　MPI 初期化
      call MPI_INIT(ierr)

c　自プロセッサIDの取得
      call MPI_COMM_RANK(MPI_COMM_WORLD, myrank, ierr)
```



コミュニケータ．互いにメッセージを送れるプロセスの集団．MPIで定義済み．

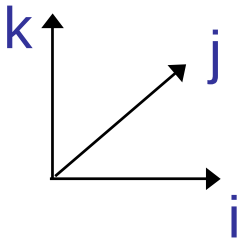
プログラミング技法②

Fortranでの配列データの並べ方

K=1,KK+1
J=1,JJ+1
I=1,II+1

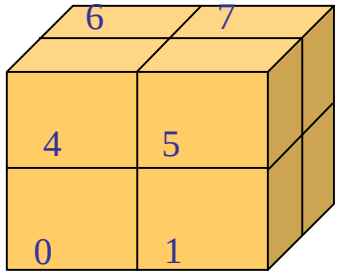
(1,1,1) (2,1,1) … (II+1,1,1) →
(1,2,1) (2,2,1) … (II+1,2,1) →
…
(1,JJ+1,1) (2,JJ+1,1) … (II+1,JJ+1,1) →

(1,1,2) (2,1,2) … (II+1,1,2) →
(1,2,2) (2,2,2) … (II+1,2,2) →
…
(1,JJ+1,2) (2,JJ+1,2) … (II+1,JJ+1,2) →



…

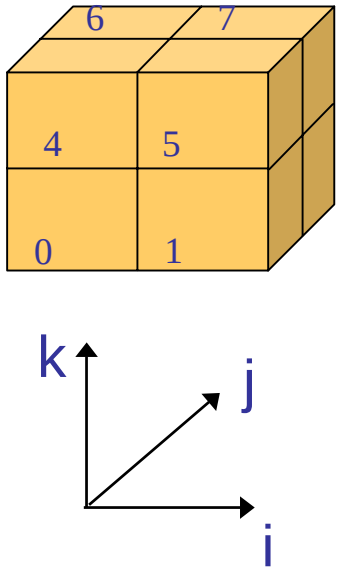
(1,1,KK+1) (2,1,KK+1) … (II+1,1,KK+1) →
(1,2,KK+1) (2,2,KK+1) … (II+1,2,KK+1) →
…
(1,JJ+1,KK+1) (2,JJ+1,KK+1) … (II+1,JJ+1,KK+1) →



プログラミング技法③

要素が配列の中のエントリである場合，MPI_TYPE_VECTORを使用して，データをまとめて送受信する（パッキング）．

MPI_TYPE_VECTOR（要素数，各要素内配列エントリ数，各要素間隔，
配列データ型，パッキングデータ名，ierr）



jkplan (I=II+1)
- 要素数：　(JJ+1)(KK+1)
- 各要素内配列エントリ数：　1
- 各要素間隔：　(II+1)

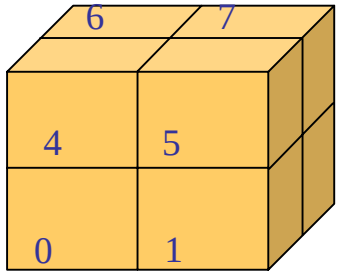
kiplan (J=JJ+1)
- 要素数：　(KK+1)
- 各要素内配列エントリ数：　(II+1)
- 各要素間隔：　(II+1)(JJ+1)

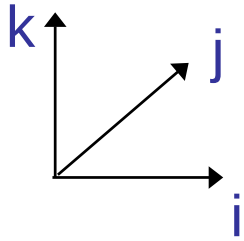
ijplan (K=KK+1)
- 要素数：　1
- 各要素内配列エントリ数：　(II+1)(JJ+1)
- 各要素間隔：　(II+1)(JJ+1)(KK+1)

プログラミング技法④

c　送受信データのパッキングのための定義

```
integer jkplan
integer kiplan
integer ijplan

 call MPI_TYPE_VECTOR(
&     3*(KK+1)*(JJ+1), 1, (II+1),
&     MPI_DOUBLE_PRECISION, jkplan, ierr)

 call MPI_TYPE_VECTOR(
&     3*(KK+1), (II+1)*1, (II+1)*(JJ+1),
&     MPI_DOUBLE_PRECISION, kiplan, ierr)

 call MPI_TYPE_VECTOR(
&     3, (JJ+1)*(II+1)*1, (II+1)*(JJ+1)*(KK+1),
&     MPI_DOUBLE_PRECISION, ijplan, ierr)

 call MPI_TYPE_COMMIT(jkplan, ierr)
 call MPI_TYPE_COMMIT(kiplan, ierr)
 call MPI_TYPE_COMMIT(ijplan, ierr)
```

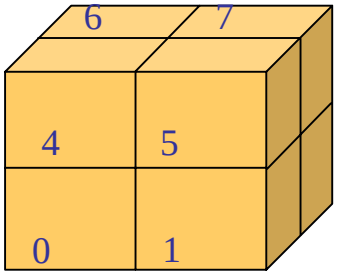


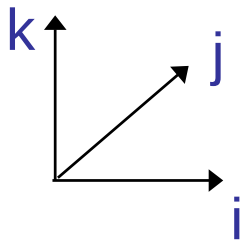


使用する前に
コミットする．
これで使用可能．

プログラミング技法⑤

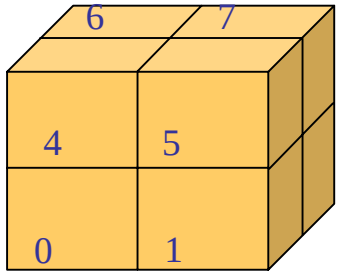
```
c       MPI 送受信 E Field
        if (mod(myrank,2).eq.1) then
c       1,3,5,7 は 0,2,4,6 に 送信
        call MPI_SEND   (E(  1,1,1,1),1, jkplan, myrank-1,
     &          tag(1), MPI_COMM_WORLD, ierr)
        else
c       0,2,4,6 は 1,3,5,7 から 受信
          call MPI_RECV(E(II+1,1,1,1),1, jkplan, myrank+1,
     &          tag(1), MPI_COMM_WORLD, status, ierr)
        endif
```

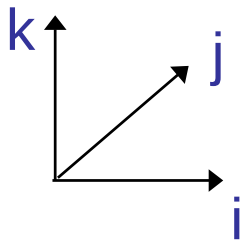




プログラミング技法⑥

```
c       MPI 送受信 H Field
        if (mod(myrank,2).eq.0) then
c         0,2,4,6 は 1,3,5,7 へ送信
          call MPI_SEND (H(II,0,0,1),1, jkplan, myrank+1,
     &         tag(1), MPI_COMM_WORLD, ierr)
        else
c         1,3,5,7 は 0,2,4,6 から 受信
          call MPI_RECV (H( 0,0,0,1),1, jkplan, myrank-1,
     &         tag(1), MPI_COMM_WORLD, status, ierr)
        endif
```

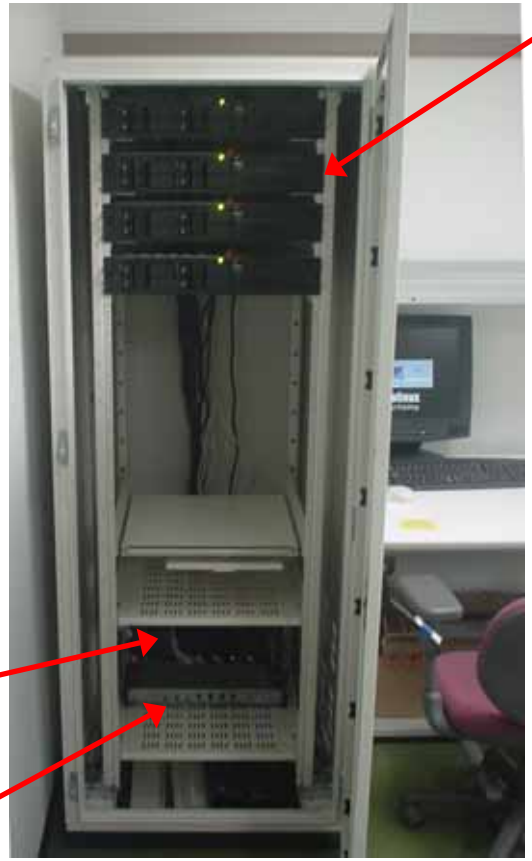




並列ＦＤＴＤシステムの構成

CPU　：　　Pentinum III 1.26GHz
並列数：　　8
メモリ：　　2GB x 8
OS　：　　TurboLinux 7
コンパイラ：　Intel Fortran (Fortran 90)
NIC：　　ギガビット

Dual CPU
PCワークステーション：4台



スイッチングハブ

ディスプレー切替え器

設定・実行メモ

●MPICHをインストール.

●MPICHで各ホストのコマンドにｒｓｈを使用.

●ネットワーク環境の利用のためnfsを立ち上げる.

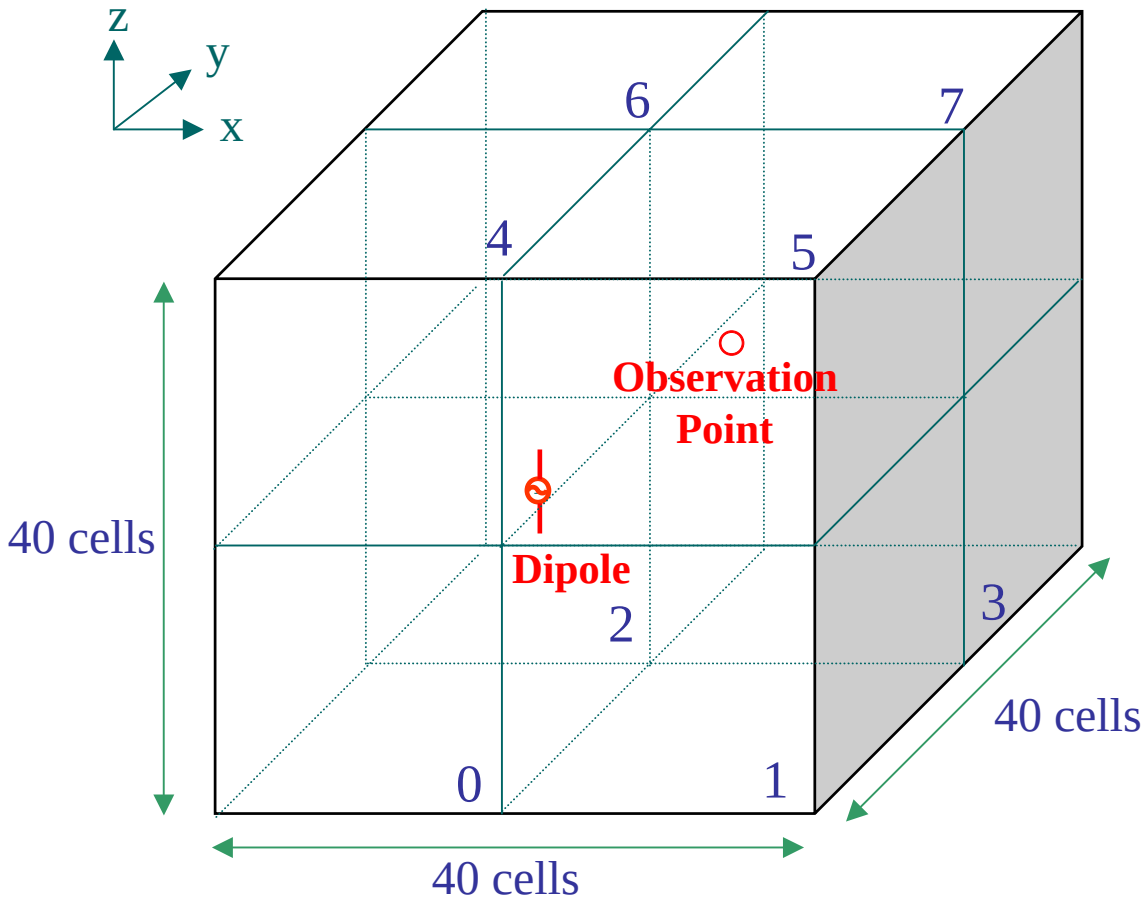
●mountの設定は /etc/fstab で行う.
　host_name1:　　/home /nfs

●コンパイル：　mpif90　filename.f

●実行：　mpirun –np 8 –nolocal a.out&

並列プログラムの検証

ガウシャンパルスで微小ダイポールアンテナを励振



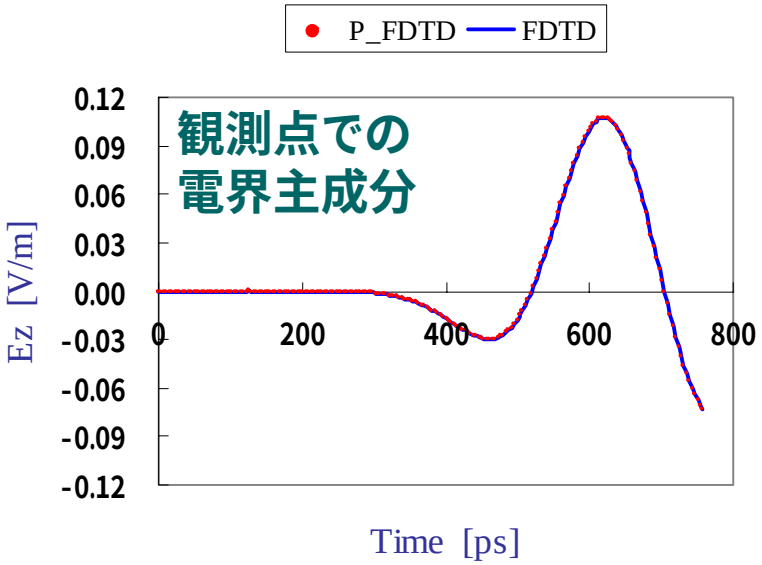
解析空間：
　４０×４０×４０セル

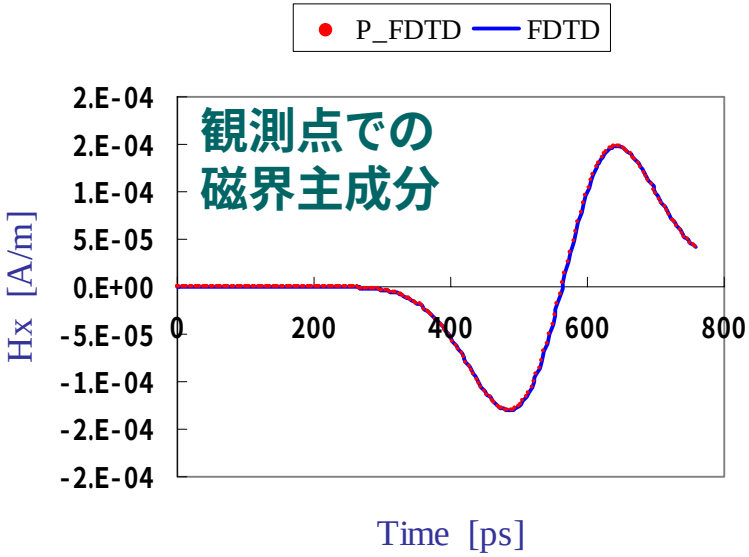
ダイポール：
　サブ空間0に配置

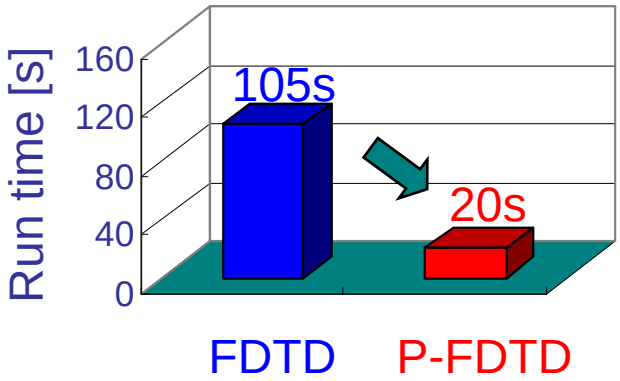
観測点O
　サブ空間７に配置

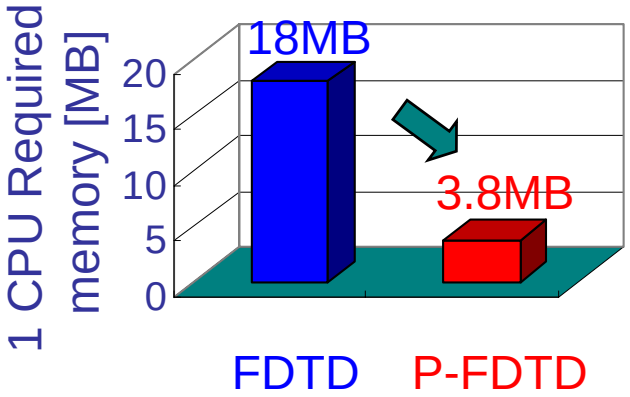
反復回数：
　1000

並列ＦＤＴＤプログラム検証結果

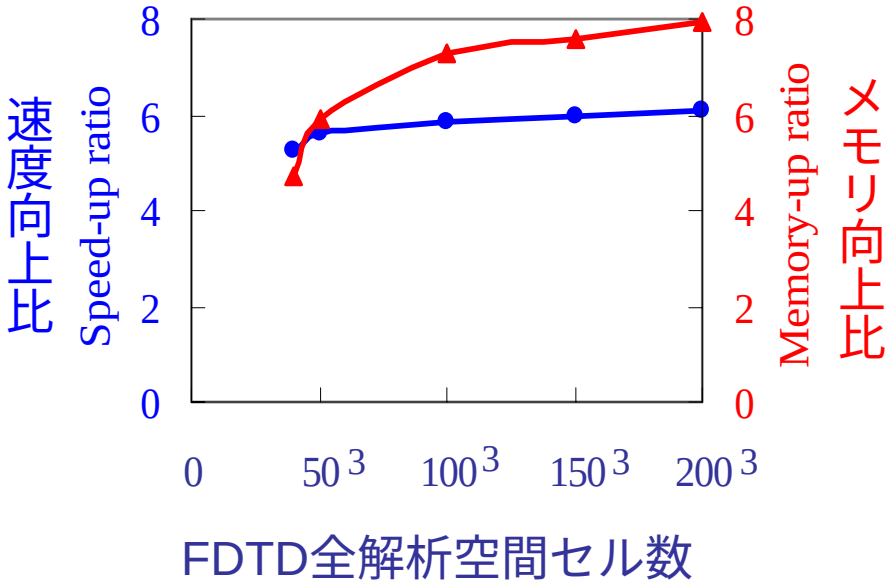








並列ＦＤＴＤシステムの性能評価



8並列で
6倍の速度向上
7.9倍のメモリ拡張

速度向上比：　非並列計算で要した時間／８並列計算で要した時間

メモリ向上比：非並列計算の所要メモリ
／８並列計算で１CPUの所要メモリ