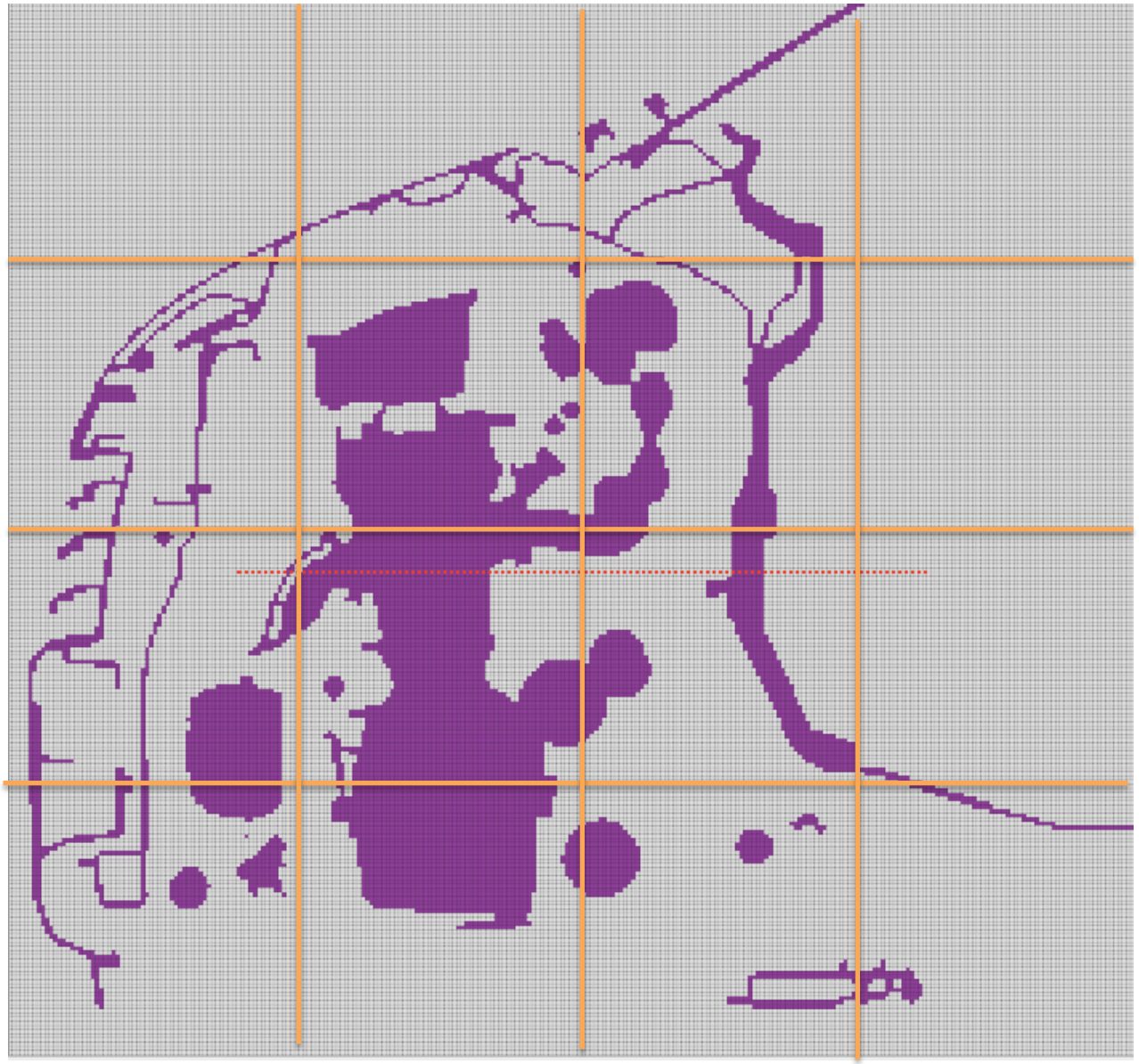
CIOlib Tutorial
Cartesian I/O library

2014-04-16

AICS, RIKEN

領域分割



TOC

- CIOlibの概要
- ビルド方法
- 基本API
- ステージング
- ファイル変換ツール

CIOlib ?

- 領域分割型の並列計算時、プロセス毎にファイルI/Oが発生
- 多数のファイルが生成される

  ↓
- どこにどのファイル？
- I/O処理

  ↓
- まず、分散並列ファイルを管理をしよう

他のツールと何が違う？

- netCDF/HDF5
  - ファイルフォーマット
  - 高性能I/O
  - 多機能データベース
- ADIOS
  - よりメタな機能を提供
  - 最適なフォーマットI/Oドライバを選択可能
- CIOlib
  - よりコンパクトな仕様（構造格子に特化）
  - 他ライブラリ、言語への依存性少ない
  - 性能は今のところ、as is

基本的な考え方

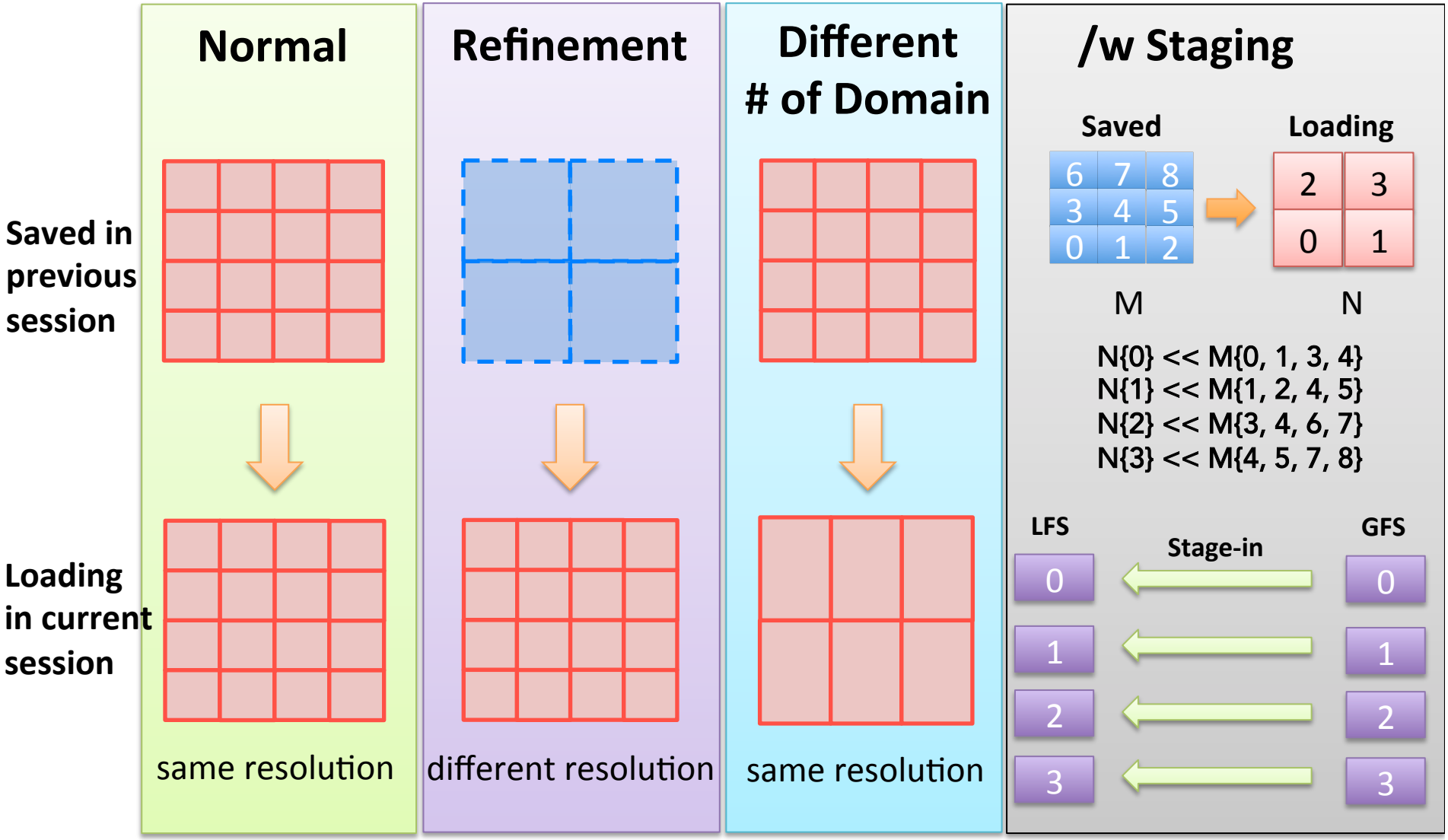
- ポータビリティ重視
  - ライブラリ依存性を少なく
  - 軽量
  - 仕様を小さく
- データ管理を主体
  - 性能面は順次対応

機能

- 直交格子データのファイル入出力管理を行う C++ クラスライブラリ

- 機能
  - DFI ファイル(メタ情報)による格子,領域分割情報の管理
  - SPH,BOVファイル形式に対応
  - MxN ロード対応(並列数が異なる場合のロード処理)
  - リファインメントロード(粗→密)対応
    - 各方向の格子数が 1/2(1/8@3 次元)の場合のロード処理
  - ステージング対応
    - 外部プログラムによる,ランク毎のディレクトリへのファイルコピー機能
  - 並列分散ファイルコンバータ対応
    - 外部プログラムによる,並列ファイル変換機能
    - SPH, BOV → SPH, BOV, PLOT3D, AVS, VTK

様々なリスタートパターンの提供

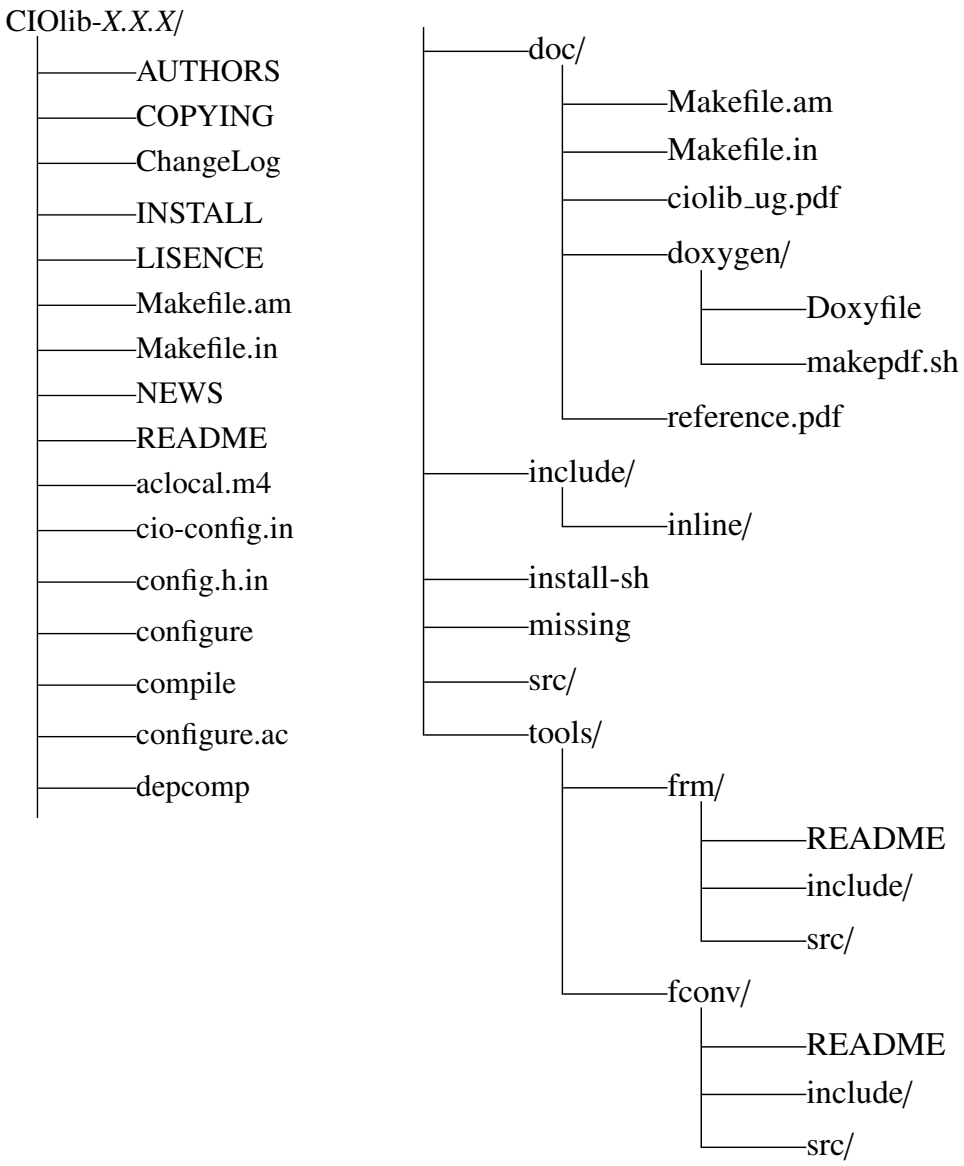


動作環境

- 動作確認済みのプラットホーム、コンパイラ
  - Mac OSX, Gnu/Intel compiler
  - Linux(Intel arch.), Gnu/Intel compiler
  - K/FX, Fujitsu compiler

- 必要なライブラリ
  - MPI library
  - TextParser
  - (CPMlib)　並列でファイルフォーマット変換時

パッケージ



https://github.com/avr-aics-riken/CIOlib/

- GNU autotoolsによるパッケージング
- frm　ファイルランクマップ
- fconv　ファイル変換

パッケージのビルド

- 簡単な手順

```
$ tar xvfz CIOlib-x.x.x.tar.gz
$ cd CIOlib-x.x.x
$ ./configure [option]
$ make
$ make install
```

ほかに、

```
$ make clean
$ make distclean
```

Configure option

--host=hostname
クロスコンパイル時のオプション
--prefix=INSTALL_DIR
インストール先
デフォルト /usr/local/CIOlib
--with-MPI=(yes|no)
並列モジュール時に指定
デフォルト yes
--with-ompi=OPENMPI_DIR
MPIライブラリの指定
--with-parser=TEXTPARSER_PATH
TextParserライブラリ指定
--with-cpm=CPM_PATH
CPMlibの指定

--with-frm=(no|yes)
frmツールのインストール指定

CXX=CXX_COMPILER

CXXFLAGS=CXX_OPTIONS

F90=FORTRAN90_COMPILER

F90FLAGS=FORTRAN90_OPTIONS

Read INSTALL

Configure example 1

```
## Intel compiler, OpenMPI.

  $ ./configure --with-ompi=/opt/openmpi \
                --with-parser=/usr/local/FFV/TextParser \
                --with-cpm=/usr/local/FFV/CPMlib \
                CXX=icpc \
                CXXFLAGS="-O3 -Wall" \
                F90=ifort \
                F90FLAGS=-O3
```

Configure example 2

```
## OpenMPI with wrapper compiler, file converter and frm.

  $ ./configure --prefix=/usr/local/FFV/CIOlib \
                --with-parser=/usr/local/FFV/TextParser \
                --with-cpm=/usr/local/FFV/CPMlib \
                --with-frm=yes \
                CXX=mpicxx \
                CXXFLAGS=-O3 \
                F90=mpif90 \
                F90FLAGS=-O3
```

Configure example 3

```
## K-computer cross-compiling

  $ ./configure --prefix=$1 \
                --with-parser=hoge \
                --with-cpm=hogehoge \
                --host=sparc64-unknown-linux-gnu \
                CXX=mpiFCCpx \
                CXXFLAGS=-Kfast \
                F90=mpifrtpx \
                F90FLAGS=-Kfast
```

Install demo

- Mac OSXへのインストール

```
$ ./configure --prefix=/usr/local/FFV/CIOlib \
              --with-parser=/usr/local/FFV/TextParser \
              --with-cpm=/usr/local/FFV/CPMlib \
              --with-frm=yes \
              CXX=mpicxx \
              CXXFLAGS=-O3 \
              F90=mpif90 \
              F90FLAGS=-O3
```

OpenMPIのラッパーコンパイラを利用

Install先のディレクトリ

```
fermium:FFV keno$ ls CIOlib/

bin/cio-config
doc/*.pdf
include/ header files
lib/libCIO.a
share/AUTHORS  COPYING  ChangeLog  LICENSE  README
```

京のフロントエンドでfrmを使う場合

Compilation of staging tool

The tool must be compiled by an native compiler on the login node. If the front-end login node does not have MPI library, specify –with-MPI=no option.

```
$ ./configure --prefix=hogehoge \
              --–with-MPI=no \
              --with-parser=hoge \
              --with-cpm=foo \
              --with-frm=yes \
              CXX=g++ \
              FC=gfortran
```

cio-configの利用方法

```
fermium:bin keno$ ./cio-config --cxx
Mpicxx

fermium:bin keno$ ./cio-config --cflags
-I/usr/local/FFV/CIOlib/include -I/usr/local/FFV/TextParser/include

fermium:bin keno$ ./cio-config --libs
-L/usr/local/FFV/CIOlib/lib -lCIO -L/usr/local/FFV/TextParser/lib –lTPmpi

fermium:bin keno$ ./cio-config --help
Usage: cio-config [OPTION]
Known values for OPTION are:
  --cxx       print C++ compiler command
  --cflags    print C/C++ pre-processor and compiler flags
  --libs      print library linking information for C++ program
  --help      display this help and exit
  --version   output version information
```

API の利用

- まず、cio_DFI.h をインクルード
  - ユーザが利用可能なAPIを定義している
  - マクロ
  - 列挙型
  - エラーコード
  - 詳細は、ユーザガイドの3.1

基本的なファイル構造

```
~/proc.dfi
  prs.dfi
  vel.dfi
  temp.dfi
  hoge/prs_0000000000_id000000.sph
       prs_...
       vel_*.id*.sph
       temp_*.id*.sph
```

dfiファイル

- 2種類のdfi (distributed file information)
  - ファイルの属性や時系列情報、アノテーション情報を記述するindex.dfi
  - 並列プロセスの情報を記述するproc.dfi

index.dfi ファイル仕様（1/5）

```
FileInfo {                              //ファイル情報
  DFIType = “Cartesian”                 //dfiの種別                    ※1
  DirectoryPath  = “./“                 //フィールドデータの存在するディレクトリ
  TimeSliceDirectory = “off”             //時刻毎のディレクトリ作成オプション
  Prefix = “vel“                        //ベースファイル名       ※2
  FileFormat  = “bov“                   //ファイルタイプ、拡張子  ※2
  FieldFilenameFormat=“step_rank”  //ファイル命名基準       ※2
  GuideCell  = 0                        //仮想セル数
  DataType  = “Float32“                 //データタイプ           ※3
  Endian = “little“                     //データのエンディアン       ※4
      :
}
```

（※1） CIOlibでは“Cartesian”のみ有効
（※2）ファイル名　“step_rank”[Prefix]_[ステップ番号:10桁]_id[RankID:6桁].[ext]
“rank_step”[Prefix]_id[RankID:6桁]_[ステップ番号.10桁].[ext]
逐次時　　[Prefix]_[ステップ番号:10桁].[ext]
（※3）Int8, UInt8, Int16, UInt16, Int32, Uint32, Int64, UInt64, Float32, Float64
（※4）little, big, 省略時:実行プラットフォームと同じ

index.dfi ファイル仕様（2/5）

```
FileInfo {                              //ファイル情報
      :
  ArrayShape = “nijk“                   //配列形状                    ※5
  Component  = 3                        //成分数（スカラーは不要）
  Variable[@] { name = “u“  }           //成分名（Component個、
  Variable[@] { name = “v“  }           //       スカラーの場合も記述）
  Variable[@] { name = “w“  }
}
FilePath {            //ファイルパス情報
  Process = “proc.dfi“      //procファイル名
}
```

（※5）　ijkn, nijk	ijkn:(imax,jmax,kmax,Component) nijk:(Component,imax,jmax,kmax)

index.dfi ファイル仕様（3/5）

```
Unit List{                  //単位系情報リスト
 Length {                   //単位系（必要に応じて追加）
     Unit = "M“                      //(NonDimensional, m, cm, mm)
     Reference= 1.0                  // 規格化に用いた長さスケール
 }
 Velocity  {
     Unit=“m/s”                      //(NonDiemsional,m/s)
     Reference=3.4                   //代表速度
  }
  Pressure  {
     Unit=“Pa”                       //(NonDimensional,Pa)
     Referencs=0.0                   //基準圧力
     Difference=510.0                //圧力差
 }
 Temperature {
      Unit=“Celsius”                 //(Nondimensional,Celsius)
      Reference=10.0                 //基準温度
   Difference=35.0                   //温度差
 }
}
```

index.dfi ファイル仕様（4/5）

```
TimeSlice {                 // 時系列データ
  Slice[@] {                // ファイル出力回数分
  Step = 0                  // 出力ステップ
    Time = 0.0              // 出力時刻
    AverageTime = 20        // 平均時間（必要に応じて出力）
    AverageStep = 30        // 平均ステップ（必要に応じて出力）

    VectorMinMax {          // u, v, w合成値のmin/max値（Component>1のとき）
      Min  = -1.56e-2               // 最小値
      Max  = 8.2e-01                // 最大値
    }
        :
```

（次頁につづく）

index.dfi ファイル仕様（5/5）

```
TimeSlice {                  //時系列データ
  Slice[@] {                 // ファイル出力回数分
        :（前頁から）
   MinMax[@] {               // Component個
      Min  = -1.56e-2        //u最小値
      Max  = 8.2e-01         //u最大値
   }
   MinMax[@] {
      Min  = -1.56e-2        //v最小値
      Max  = 8.2e-01         //v最大値
   }
   MinMax[@] {
       Min  = -1.56e-2       //w最小値
      Max  = 8.2e-01         //w最大値
   }
    ... 任意のアノテーションを追加可能
  }

   Slice[@] {
     :
```

proc.dfi ファイル仕様（1/2）

```
Domain {                                    // ドメイン情報
 GlobalOrigin  = (3.0, -3.0, -3.0)// 計算空間の起点座標
 GlobalRegion  = (6.0,  6.0, 6.0) // 計算空間の各軸方向の長さ
 GlobalVoxel   = (64, 64, 64)     // 計算領域全体のボクセル数
 GlobalDivision= (1, 1, 1)        // 計算領域の部分領域の分割数
}


MPI {                                   //並列情報
  NumberOfRank  = 128                   // プロセス数
  NumberOfGroup = 1                     // グループ数
}
```

proc.dfi ファイル仕様（2/2）

```
Process {
  Rank[@] {                    // NumberOfRank個
    ID         = 0               // ランク番号
      HostName  = "Strontium“   // ホストノード名(必須ではない)
      VoxelSize = (64, 64, 64) // ボクセルサイズ
      HeadIndex = (1, 1, 1)     // 始点インデクス
                                // グローバルで(1,1,1)からスタート
    TailIndex = (64, 64, 64)    // 終点インデクス
                                // グローバルで(64, 64, 64)が終端
 }

 Rank[@] {
   ID  = 1
   ...
 }
 ...
}
```

フィールドファイルフォーマット

1.SPH形式

V-Toolsにおける計算結果を格納するバイナリ形式ファイル

データ属性、ボクセルサイズ、原点座標、ボクセルピッチ、時刻、
データレコードが順に1レコードずつ格納されている
次項SPHファイルレコード形式参照

2.BOV形式

可視化ソフトウェア「VisIt」のBrick of Values形式ファイル

データ配列のみが単純に格納されている
次項BOVファイルレコード形式参照

SPHファイルレコード形式（1/8）

レコード名	意味
データ属性レコード	データの属性を記述する （データ属性レコード参照）
ボクセルサイズレコード	ボクセルサイズを記述する （ボクセルサイズレコード参照）
原点座標レコード	原点座標を記述する （原点座標レコード参照）
ボクセルピッチレコード	ボクセルピッチを記述する （ボクセルピッチレコード参照）
時刻レコード	タイムステップと時刻を記述する （時刻レコード参照）
データレコード	データを記述する （データレコード参照）

SPHファイルレコード形式（2/8）

データ属性レコード

名称	表現	サイズ	説明
Size	整数	4 byte	レコード長(=8) (※1)
svType	整数	4 byte	データ種別フラグ (※2)
dType	整数	4 byte	データ型フラグ (※3)
Size	整数	4 byte	レコード長(=8) (※1)

(※1)Fortranの書式なし出力の形式に合わせた項目で、データレコード長のバイト数でデータを
　　はさむ形式をとる
(※2)スカラーデータかベクトルデータかを判断するフラグ
　　１：スカラー、２：ベクトル
(※3)記述されるデータの型（単精度/倍精度）を判断するフラグ
　　１：単精度、２：倍精度

SPHファイルレコード形式（3/8）

ボクセルサイズレコード

名称	表現	サイズ	説明
Size	整数	4 byte	レコード長(=12or24)（※2)
IMAX	整数	4or8 byte(※1)	I方向ボクセル数
JMAX	整数	4or8 byte(※1)	J方向ボクセル数
KMAX	整数	4or8 byte(※1)	K方向ボクセル数
Size	整数	4 byte	レコード長(=12or24)（※2)

(※1)データ型フラグ(dType)の値(単精度/倍精度)により異なる
単精度(dType=1):4byte、倍精度(dType=2):8byte

(※2)データ型フラグ(dType)の値(単精度/倍精度)により異なる
単精度(dType=1):12byte、倍精度(dType=2):24byte

SPHファイルレコード形式（4/8）

原点座標レコード

名称	表現	サイズ	説明
Size	整数	4 byte	レコード長(=12or24)　(※2)
XORG	整数	4or8 byte(※1)	X軸方向原点座標値
YORG	整数	4or8 byte(※1)	Y軸方向原点座標値
ZORG	整数	4or8 byte(※1)	Z軸方向原点座標値
Size	整数	4 byte	レコード長(=12or24)　(※2)

(※1)データ型フラグ(dType)の値(単精度/倍精度)により異なる
　　単精度(dType=1):4byte、倍精度(dType=2):8byte

(※2)データ型フラグ(dType)の値(単精度/倍精度)により異なる
　　単精度(dType=1):12byte、倍精度(dType=2):24byte

SPHファイルレコード形式（5/8）

ボクセルピッチレコード

名称	表現	サイズ	説明
Size	整数	4 byte	レコード長(=12or24)　(※2)
XPITCH	整数	4or8 byte(※1)	X方向ボクセルピッチ
YPITCH	整数	4or8 byte(※1)	Y方向ボクセルピッチ
ZPITCH	整数	4or8 byte(※1)	Z方向ボクセルピッチ
Size	整数	4 byte	レコード長(=12or24)　(※2)

(※1)データ型フラグ(dType)の値(単精度/倍精度)により異なる
　　単精度(dType=1):4byte、倍精度(dType=2):8byte

(※2)データ型フラグ(dType)の値(単精度/倍精度)により異なる
　　単精度(dType=1):12byte、倍精度(dType=2):24byte

SPHファイルレコード形式（6/8）

時刻レコード

名称	表現	サイズ	説明
Size	整数	4 byte	レコード長(=8or12)　(※2)
STEP	整数	4or8 byte(※1)	タイムステップ
TIME	整数	4or8 byte(※1)	時刻
Size	整数	4 byte	レコード長(=8or12)　(※2)

(※1)データ型フラグ(dType)の値(単精度/倍精度)により異なる
単精度(dType=1):4byte、倍精度(dType=2):8byte

(※2)データ型フラグ(dType)の値(単精度/倍精度)により異なる
単精度(dType=1):12byte、倍精度(dType=2):24byte

SPHファイルレコード形式（7/8）

SPHデータレコード（svType=1:スカラー）

名称	表現	サイズ	説明
Size	整数	4 byte	レコード長(※2)
DATA(0,0,0)	実数	4or8 byte(※1)	格子点(0,0,0)のデータ値
DATA(1,0,0)	実数	4or8 byte(※1)	格子点(1,0,0)のデータ値
DATA(2,0,0)	実数	4or8 byte(※1)	格子点(2,0,0)のデータ値
・・・			
DATA(IMAX-1,JMAX-1,KMAX-1)	実数	4or8 byte(※1)	格子点(IMAX-1,JMAX-1,KMAX-1)のデータ値
Size	整数	4 byte	レコード長(※2)

(※1)データ型フラグ(dType)の値(単精度/倍精度)により異なる

単精度(dType=1):4byte、倍精度(dType=2):8byte

(※2)データ型フラグ(dType)の値(単精度/倍精度)により異なる

単精度(dType=1):IMAX × JMAX × KMAX × 4　byte

倍精度(dType=2):IMAX × JMAX × KMAX × 8　byte

SPHファイルレコード形式（8/8）

SPHデータレコード（svType=2:ベクトル）

名称	表現	サイズ	説明
Size	整数	4 byte	レコード長(※2)
U(0,0,0)	実数	4or8 byte(※1)	格子点(0,0,0)のUデータ値
V(0,0,0)	実数	4or8 byte(※1)	格子点(0,0,0)のVデータ値
W(0,0,0)	実数	4or8 byte(※1)	格子点(0,0,0)のWデータ値
U(1,0,0)	実数	4or8 byte(※1)	格子点(1,0,0)のUデータ値
V(1,0,0)	実数	4or8 byte(※1)	格子点(1,0,0)のVデータ値
W(1,0,0)	実数	4or8 byte(※1)	格子点(1,0,0)のWデータ値
・・・			
U(IMAX-1,JMAX-1,KMAX-1)	実数	4or8 byte(※1)	格子点(IMAX-1,JMAX-1,KMAX-1)のUデータ値
V(IMAX-1,JMAX-1,KMAX-1)	実数	4or8 byte(※1)	格子点(IMAX-1,JMAX-1,KMAX-1)のVデータ値
W(IMAX-1,JMAX-1,KMAX-1)	実数	4or8 byte(※1)	格子点(IMAX-1,JMAX-1,KMAX-1)のWデータ値
Size	整数	4 byte	レコード長(※2)

(※1)データ型フラグ(dType)の値(単精度/倍精度)により異なる
　　単精度(dType=1):4byte、倍精度(dType=2):8byte
(※2)データ型フラグ(dType)の値(単精度/倍精度)により異なる
　　単精度(dType=1):IMAX × JMAX × KMAX × 4 × 3 byte
　　倍精度(dType=2):IMAX × JMAX × KMAX × 8 × 3 byte

BOVファイルレコード形式（1/2）

ijkn配列　v(i,j,k,n)の記述例

配列要素	説明
v(0,0,0,0)	格子点 (0,0,0)の成分0のデータ値
v(1,0,0,0)	格子点 (1,0,0)の成分0のデータ値
・・・	
v(imax-1,jmax-1,kmax-1,0)	格子点 (imax-1,jmax-1,kmax-1)の成分0のデータ値
v(0,0,0,1)	格子点 (0,0,0)の成分1のデータ値
・・・	
v(imax-1,jmax-1,kmax-1,n-1)	格子点 (imax-1,jmax-1,kmax-1)の成分n-1のデータ値

BOVファイルレコード形式（2/2）

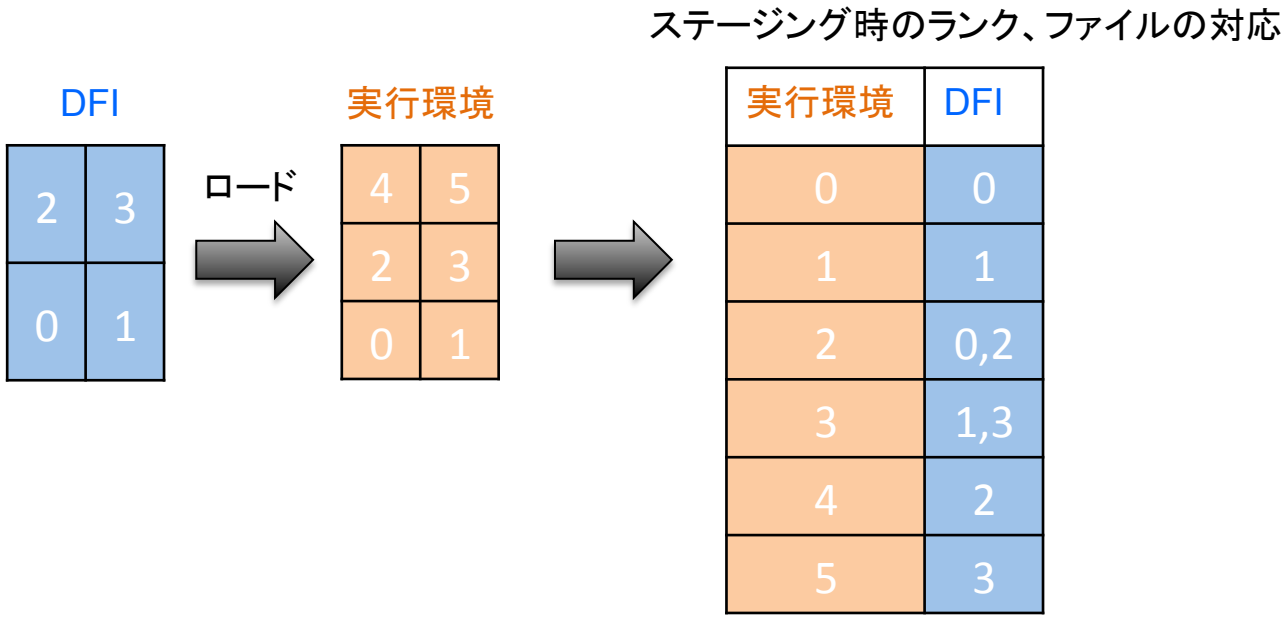
nijk配列　v(n,i,j,k)の記述例

配列要素	説明
v(0,0,0,0)	格子点 (0,0,0)の成分0のデータ値
v(1,0,0,0)	格子点 (0,0,0)の成分1のデータ値
…	
v(n-1,0,0,0)	格子点 (0,0,0)の成分n-1のデータ値
v(0,1,0,0)	格子点 (1,0,0)の成分0のデータ値
…	
v(n-1,imax-1,jmax-1,kmax-1)	格子点 (imax-1,jmax-1,kmax-1)の成分n-1のデータ値

ステージング対応(1/7)

1．機能概要

大規模クラスタ環境等におけるファイル転送の補助機能として、ランク毎に必要なファイルをランク番号で命名したディレクトリにコピーする機能



ステージング対応(2/7)

２．入力情報

- DFIファイル
- 実行プログラムに関する”Domain”相当の情報（領域分割情報ファイル）
- （オプション※１）実行プログラムにおける、ランクマップ情報（領域分割情報ファイル)
- （オプション※２）実行プログラムのhead/tail情報（領域分割情報ファイル）
- ランク毎のディレクトリ命名のPrefix

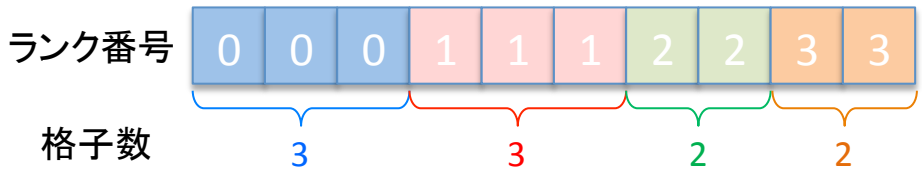
（※１）デフォルト　I→J→K、領域分割情報ファイルのランクマップがある時はそれに従う

（※２）デフォルトはCPMライブラリの自動領域分割、領域分割情報ファイルのHeadIndex、TailIndex
がある時はそれに従う

⇒ある方向について格子数NV、領域分割数ND（ランク番号0～ND-1)とした時、あるランクにおける格子数は int(NV/ND)とする

ただし、ランク番号＜NV%NDのランクの格子数は＋１とする

（例）格子数10、領域分割数4の例



ステージング対応(3/7)

- FCONVの入力ファイル
  FCONVの入力ファイル仕様はFCONVファイルのマニュアル参照
- FCONVをMx1，MxMで実行するときの実行並列数

(※3)領域分割情報ファイルとFCONV入力ファイルの併用は不可

(※4)FCONV入力ファイルが指定されているときはFCONV入力ファイル内で
指定されたDFIファイルが有効になる.

(※5)FCONVの実行並列数が指定されない場合は実行並列数を1とする

(※6)MxNでFCONVを実行する場合は実行並列数はFCONVの入力ファイル
で指定されている分割情報から求められる.

ステージング対応(4/7)

3. 領域分割情報ファイル

実行プログラムにおける領域分割の情報を記述したファイル。ステージング対応機能で使用する。

```
Domain {
 GlobalVoxel = (64, 64, 64)//計算領域全体のボクセル数
 GlobalDivision = (1, 1, 1)//計算領域の部分領域の分割数
 ActiveSubdomainFile = “ファイル名“ //ActiveSubdomainファイル名
}
MPI {
  NumberOfRank = 1          // プロセス数
}
Process {
  Rank[@] {                 // NumberOfRank個
   ID = 0              //ランク番号
   VoxelSize = (64, 64, 64)//ボクセルサイズ
   HeadIndex = (1, 1, 1)          //始点インデクス
   TailIndex = (64, 64, 64)//終点インデクス
  }
}
```

- Domainタグは必須
  ただし、ActiveSubdomainFileは任意
  FCONVで入力領域が指定されているときは、無効
  GlobalDivisionはMxNのみ有効
- MPI、Processタグは任意
  ランクの配置方向が I→J→Kでない配置の場合、もしくはHeadIndex、TailIndex の位置が異なる場合に記述
  FCONVで入力領域が指示されているときは無効
  - ◆ HeadIndex、TailIndexが異なる時、ランク配置方向が I→J→Kでないとき、HeadIndex、TailIndexを記述

ステージング対応(5/7)

4. 実行例

< 4分割(2,1,2)の結果を8分割(2,2,2)でリスタートする例 >

・ソルバーのDomain情報格納ファイル
(solvproc.txt)

```
Domain{
  GlobalVoxel=(64,64,64)
  GlobalDivision=(2,2,2)
  ActiveSubdomainFile=“”
}
```

・振り分け対象ステップ番号

100step

・出力ディレクトリ

hoge/

・実行コマンド

$ frm –i solvproc.txt –s 100 –o hoge old/prs.dif old/vel.dfi

・振り分け対象のファイル

oldディレクトリ配下のprs.dfi,vel.dfi
実態のsphファイルはSPH/ディレクトリに存在

```
old/
    prs.dfi
    vel.dfi
    proc.dfi
    SPH/
        prs_0000000000_id000000.sph
        prs_0000000000_id000001.sph
        prs_0000000000_id000002.sph
        prs_0000000000_id000003.sph
        prs_0000000100_id000000.sph
        prs_0000000100_id000001.sph
        prs_0000000100_id000002.sph
        prs_0000000100_id000003.sph
        vel_0000000000_id000000.sph
        vel_0000000000_id000001.sph
        vel_0000000000_id000002.sph
        vel_0000000000_id000003.sph
        vel_0000000100_id000000.sph
        vel_0000000100_id000001.sph
        vel_0000000100_id000002.sph
        vel_0000000100_id000003.sph
```

ステージング対応(6/7)

・実行結果
hogeディレクトリが生成され、その配下に6桁のランク番号ディレクトリが生成され、各ランク用ディレクトリ配下にそれぞれ必要なファイルがコピーされる。

hoge/000000/

prs.dfi ← DirectoryPath="./"
prs_0000000100_id000000.sph
prs_proc.dfl ← proc.dfiからコピーされる
vel.dfi ← DirectoryPath="./"
vel_0000000100_id000000.sph
vel_proc.dfi ← proc.dfiからコピーされる

hoge/000001/

prs.dfi
prs_0000000100_id000001.sph
prs_proc.dfl
vel.dfi
vel_0000000100_id000001.sph
vel_proc.dfi

hoge/000002/

prs.dfi
prs_0000000100_id000000.sph
prs_proc.dfl
vel.dfi
vel_0000000100_id000000.sph
vel_proc.dfi

hoge/000003/

prs.dfi
prs_0000000100_id000001.sph
prs_proc.dfl
vel.dfi
vel_0000000100_id000001.sph
vel_proc.dfi

ステージング対応(7/7)

hoge/000004/

prs.dfi
prs_0000000100_id000002.sph
prs_proc.dfl
vel.dfi
vel_0000000100_id000002.sph
vel_proc.dfi

hoge/000005/

prs.dfi
prs_0000000100_id000003.sph
prs_proc.dfl
vel.dfi
vel_0000000100_id000003.sph
vel_proc.dfi

hoge/000006/

prs.dfi
prs_0000000100_id000002.sph
prs_proc.dfl
vel.dfi
vel_0000000100_id000002.sph
vel_proc.dfi

hoge/000007/

prs.dfi
prs_0000000100_id000003.sph
prs_proc.dfl
vel.dfi
vel_0000000100_id000003.sph
vel_proc.dfi