

FrontISTR ビルド方法の紹介

帝京大学 戦略的イノベーション研究センター
小川 道夫

2017年11月10日(金)

目次

1. FrontISTR V4.6ビルドのための準備
2. ライブラリのビルド
3. FrontISTR V4.6のビルド
4. コツやエラー対処など
5. FrontISTR V5.0で導入される新しいビルド方法の紹介

FrontISTR V4.6 ビルドのための準備

CentOS7.3を例として説明をします

FrontISTRのビルド手順

環境設定・バイナリのインストール

OSのインストール

C/Fortranのインストール

make/CMakeのインストール

MPIライブラリのインストール



外部ライブラリのビルド・インストール

BLAS/LAPACK

metis

ScaLAPACK

MUMPS

Trilinos ML



FrontISTRのビルド・インストール

Makefile.confの設定

setup.shの実行

make/make install

方針を決める

必要な機能は何か？

- MPI, OpenMP, REVOCAP_Refiner, Metis, MUMPS, Trilinos ML?

コンパイラは何を使うか？

- GNU(gcc/gfortran) or Intel(icc/ifort)?

MPIライブラリは何を使うか？

- OpenMPI or Intel MPI?

BLAS/LAPACKは何を使うか？

- OpenBLAS or Intel MKL(Free版?コマーシャル版?)?

ここでは、

- OSは、CentOS 7.3
- コンパイラは、GNU(gcc/gfortran)
- MPIは、OpenMPI
- OpenMPを有効
- BLAS/LAPACKはOpenBLAS
- ライブラリはスタティックライブラリとしてビルド
- FrontISTRで有効にする機能
 - tools (領域分割ツールなど)
 - LAPACK
 - refiner
 - metis
 - MUMPS
 - ML

環境構築

- CentOSのイメージをダウンロードし、インストールして下さい。
- インストール後、以下のコマンドを実行し、基本的な開発環境を用意してください。
- OpenMPIとcmakeをインストールしてください。

```
$ su  
# yum group mark install "Development Tools"  
# yum update
```

```
$ su  
# yum install openmpi-devel cmake  
# exit  
$ cd $HOME  
$ vi .bash_profile  
module pruge  
module load mpi/openmpi-x86_64
```

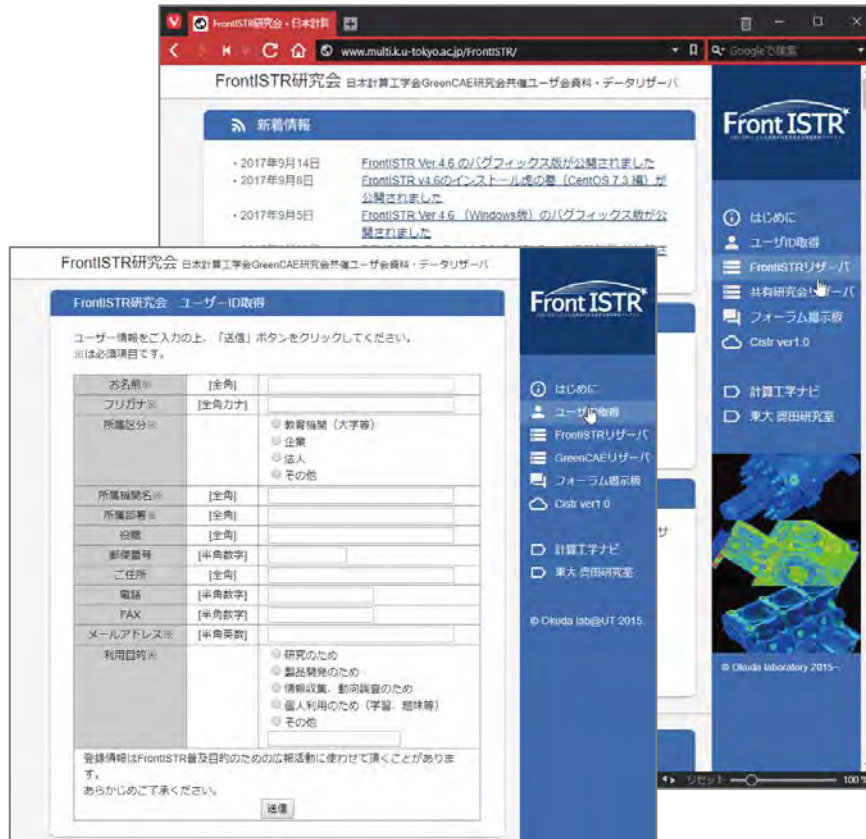
スーパーユーザの権限が必要
(管理者に相談してください)



ソフトウェアの入手

名前	リンク	備考
OpenBLAS-0.2.20	http://www.openblas.net	BLAS/LAPACK
metis-5.1.0	http://glaros.dtc.umn.edu/gkhome/metis/metis/download	領域分割
scalapack-2.0.2	http://www.netlib.org/scalapack/	MUMPSで利用
MUMPS_5.1.2	http://mumps.enseeiht.fr/	要ユーザ登録
trilinos-12.12.1	https://trilinos.org/download/	要ユーザ登録
REVOCAP_Refiner-1.1.04	http://www.multi.k.u-tokyo.ac.jp/FrontISTR	FrontISTRリザーバ
FrontISTR_V46	http://www.multi.k.u-tokyo.ac.jp/FrontISTR	FrontISTRリザーバ

FrontISTRのダウンロード



FrontISTRとREVOCAP_Refinerは

FrontISTR研究会

◦ <http://www.multi.k.u-Tokyo.ac.jp/FrontISTR/>

の「FrontISTRリザーバ」からダウンロードしてください。

初めての方は「ユーザーID取得」から登録をしてください。

ライブラリのビルド

FrontISTRで使用するライブラリのビルド方法をご紹介します

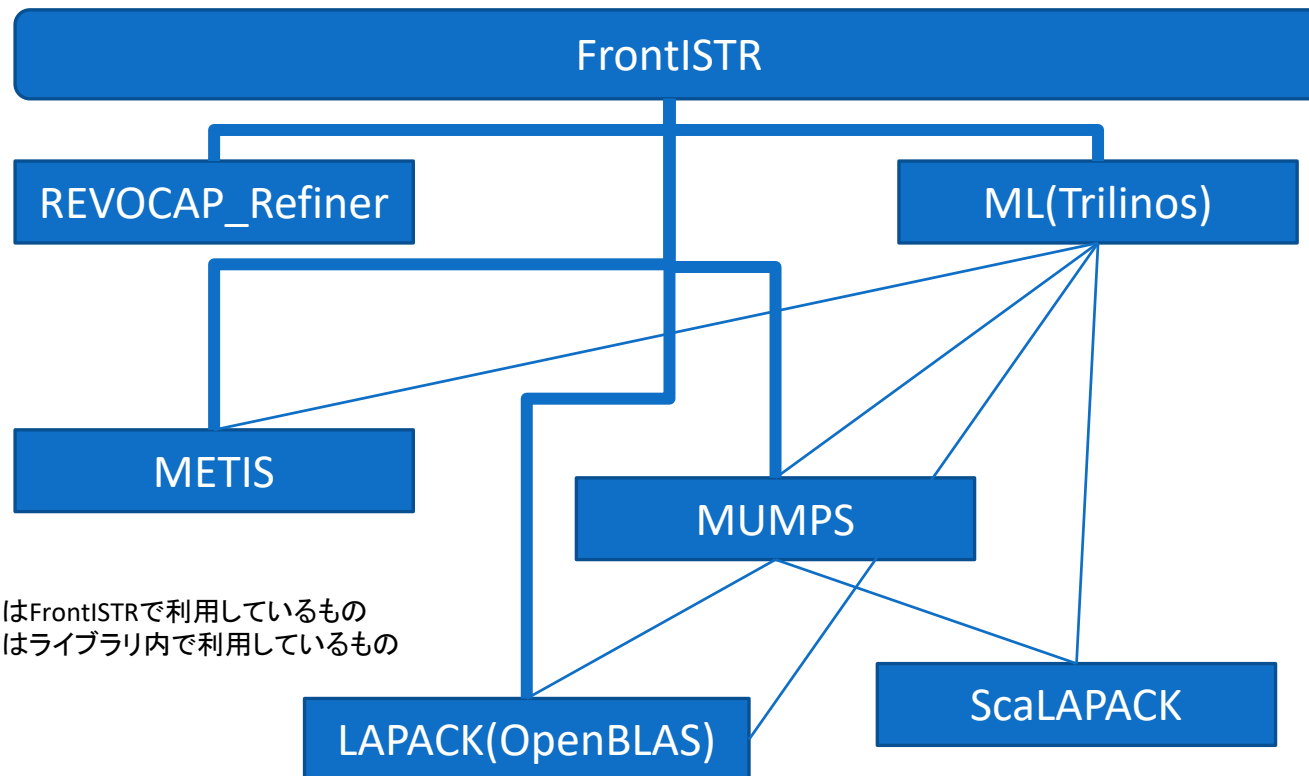
ライブラリの依存関係

➤ 依存関係を考慮してライブラリを構築する

1. LAPACK(OpenBLAS)
2. METIS
3. ScaLAPACK
4. MUMPS
5. ML(Trilinos)
6. REVOCAP_Refiner
7. FrontISTR

の順に構築していく。

- 太線はFrontISTRで利用しているもの
- 細線はライブラリ内で利用しているもの



OpenBLASのビルド

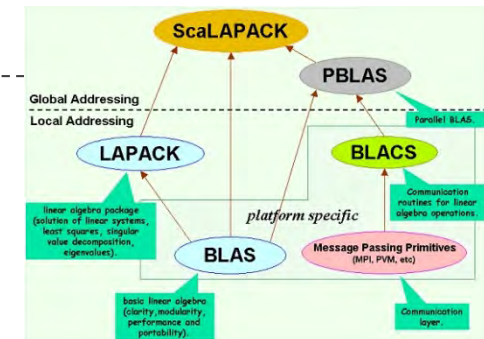
- OpenBLASは高速なフリーのBLAS/LAPACKの実装です。
- OpenMPを有効にしたライブラリを構築してください。

```
$ tar xvf OpenBLAS-0.20.0.tar.gz  
$ cd OpenBLAS-0.20.0  
$ make BINARY=64 NO_SHARED=1 USE_OPENMP=1  
$ make PREFIX=$HOME/.local install
```

4コアのマシンであれば、-j4 を付けるとコンパイル時間が節約できます。

構築したライブラリは、\$HOME/.localをインストールルートとすると後々楽です。

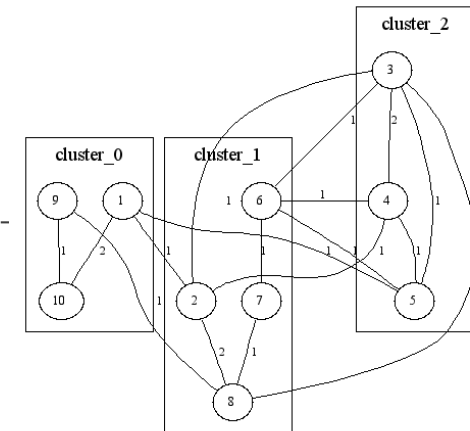
実行マシンとコンパイルマシンが違う場合、DYNAMIC_ARCH=1 を付けてください。ただし、コンパイル時間が大幅に増加します。



METISのビルド

- METISは領域分割ツールhecmw_part1で利用されます。
- Version5系列とVersion4系列がありますが、ここではVersion5系列での説明をします。
- 構築には `cmake` が必要です。

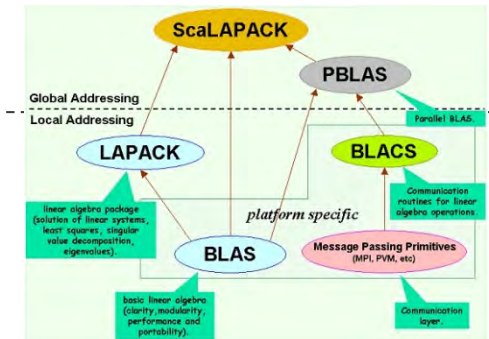
```
$ tar xvf metis-5.1.0.tar.gz  
$ cd metis-5.1.0  
$ make config prefix=$HOME/.local openmp=1 cc=gcc  
$ make  
$ make install
```



ScaLAPACKのビルド

- ScaLAPACKはMUMPSで使われています。
- FrontISTRでは直接使われていませんが、ライブラリをリンクする必要があります。
- 構築には cmake が必要です。

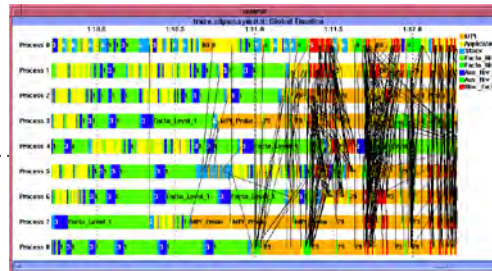
```
$ tar xvf scalapack-2.0.2.tar.gz
$ cd scalapack-2.0.2
$ mkdir build
$ cd build
$ cmake -DCMAKE_INSTALL_PREFIX=$HOME/.local -DCMAKE_EXE_LINKER_FLAGS="-fopenmp" ¥
        -DBLAS_LIBRARIES=$HOME/.local/lib/libopenblas.a ¥
        -DLAPACK_LIBRARIES=$HOME/.local/lib/libopenblas.a
        ..
$ make
$ make install
```



MUMPSのビルド

- FrontISTRでは、直接法ソルバーとしてMUMPSを利用できます。
.cntファイルの中で指定が可能です。
- Make.inc/Makefile.inc.generic をひな形として、編集します。
- 出来上がったライブラリとヘッダファイルを手動でコピーしてください。

```
$ tar xvf MUMPS_5.1.2.tar.gz  
$ cd MUMPS_5.1.2  
$ cp Make.inc/Makefile.inc.generic Makefile.inc  
$ make  
$ cp lib/*.a $HOME/.local/lib  
$ cp include/*.h $HOME/.local/include
```



```
LMETISDIR = $(HOME)/.local  
  
IMETIS = -I$(LMETISDIR)/include  
  
LMETIS = -L$(LMETISDIR)/lib -lmetis  
  
ORDERINGSF = -Dmetis -Dpard  
  
CC = mpicc -fopenmp  
  
FC = mpifort -fopenmp  
  
FL = mpifort -fopenmp  
  
LAPACK =  
  
SCALAP = -L$(HOME)/.local/lib -lscalapack  
  
INCPAR =  
  
LIBPAR = $(SCALAP)  
  
LIBBLAS = -L$(HOME)/.local/lib -lopenblas
```

Trilinos MLのビルド

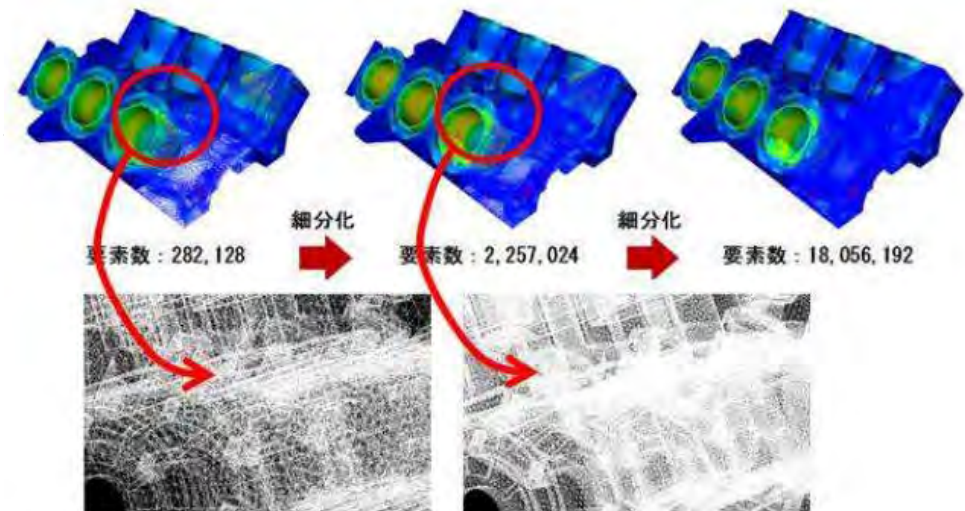
- Trilinos MLは、FrontISTRの前処理に用います。.cntファイルの中で指定が可能です。
- 構築には cmake が必要です。

```
$ tar xvf trilinos-12.12.1-Source.tar.gz
$ cd trilinos-12.12.1-Source
$ mkdir build
$ cd build
$ cmake -DCMAKE_INSTALL_PREFIX=$HOME/.local -DTPL_ENABLE_MPI=ON -DParMETIS_INCLUDE_DIRS=$HOME/local/include ¥
-DParMETIS_LIBRARY_DIRS=$HOME/local/lib -DParMETIS_LIBRARY_NAMES="parmetis;metis" -DTPL_ENABLE_ParMETIS=ON ¥
-DTPL_METIS_INCLUDE_DIRS=$HOME/local/include -DTPL_METIS_LIBRARIES=$HOME/local/lib/libmetis.a ¥
-DTPL_ENABLE_METIS=ON -DTPL_BLAS_LIBRARIES=$HOME/local/lib/libopenblas.a ¥
-DTPL_LAPACK_LIBRARIES=$HOME/local/lib/libopenblas.a -DTPL_ENABLE_LAPACK=ON -DTPL_ENABLE_SCALAPACK=ON ¥
-DMUMPS_INCLUDE_DIRS=$HOME/local/include -DMUMPS_LIBRARY_DIRS=$HOME/local/lib ¥
-DMUMPS_LIBRARY_NAMES="dmumps;mumps_common;pord" -DTPL_ENABLE_MUMPS=ON ¥
-DTrilinos_ENABLE_ML=ON -DTrilinos_ENABLE_Zoltan=ON -DTrilinos_ENABLE_Amesos=ON ¥
-DTrilinos_ENABLE_OpenMP=ON -DTrilinos_ENABLE_ALL_OPTIONAL_PACKAGES=OFF ..
$ make && make install
```

REVOCAP_Refinerのビルド

➤ REVOCAP_Refinerは、メッシュの細分化に用います。hecmw_ctrl.datの中で指定が可能です。

```
$ tar xvf REVOCAP_Refiner-1.1.04.tar.gz  
$ cd REVOCAP_Refiner-1.1.04  
$ make
```





FrontISTR V4.6 のビルド

FrontISTR本体のビルド方法を説明します

FrontISTRのビルド

- FrontISTRのアーカイブにある Makefile.conf.org を元にコンパイルに必要な情報を設定する。
- setup.shを実行し、Makefileを作成しビルドする。

```
$ tar xvf FrontISTR_V46.tar.gz
$ cd FrontISTR_V46
$ cp Makefile.conf.org Makefile.conf
$ vi Makefile.conf
$ ./setup.sh -p --with-tools --with-refiner --with-lapack ¥
--with-metis --with-mumps --with-ml
$ make
$ make install
```

```
#####
#
# Setup Configuration File for FrontISTR
#
#####

# MPI
MPI = /usr/lib64/openmpi
MPIBIN = $(MPI)/bin
MPILIBDIR = $(MPI)/lib
MPIINC = $(MPI)/include
MPILIBS = -lmpi -lmpi_cxx -lmpi_mpifh

# for install option only
PREFIX = $(HOME)/FrontISTR
BINDIR = $(PREFIX)/bin
LIBDIR = $(PREFIX)/lib
INCLUDEDIR = $(PREFIX)/include

# Metis
METISDIR = $(HOME)/local
METISLIBDIR = $(METISDIR)/lib
METISINC = $(METISDIR)/include
HEDM_METIS_VER = 5

# ParMetis
PARMETISDIR = $(HOME)/ParMetis-3.1
PARMETISLIBDIR = $(PARMETISDIR)
PARMETISINC = $(PARMETISDIR)/ParMETISlib

# Refiner
REFINER = $(HOME)/Software/REVOCAP_Refiner-1.1.04
REFINERINC = $(REFINER)/refiner
REFINERLIBDIR = $(REFINER)/lib/x86_64-linux

# Coupler
REVOCAPDIR = $(HOME)/REVOCAP_Coupler
REVOCAPINC = $(REVOCAPDIR)/librcap
REVOCAPLIBDIR = $(REVOCAPDIR)/librcap

# MUMPS
MUMPSDIR = $(HOME)/local
MUMPSINC = $(MUMPSDIR)/include
MUMPSLIBDIR = $(MUMPSDIR)/lib
MUMPSLIBS = -ldumps -lmumps_common -lpord -L$(HOME)/local/lib -lscaLapack

# MKL PARDISO
MKLDIR = $(HOME)/
MKLINC = $(MKLDIR)/include
MKLLIBDIR = $(MKLDIR)/lib

# ML
MLDIR = $(HOME)/local
MLINC = $(MLDIR)/include
MLLIBDIR = $(MLDIR)/lib

# C compiler settings
CC = mpicc -fopenmp
CFLAGS = -stdc++ -lm
CXXFLAGS = -O3

# C++ compiler settings
CXX = mpicxx -fopenmp
CXXFLAGS = -stdc++ -lm
CXXLIBS = -O3

# Fortran compiler settings
F90 = mpi90 -fopenmp
F90FLAGS = -stdc++ -L$(HOME)/local -lopenblas
F90OPTFLAGS = -O2
F90FPP = -cpp
F90LINKER = mpi90 -fopenmp

MAKE = make
AR = ar ruv
MV = mv -f
```

Makefile.conf

```
#####
#
#   Setup Configuration File for FrontISTR
#
#####

# MPI
MPIDIR      = /usr/lib64/openmpi
MPIBINDIR   = $(MPIDIR)/bin
MPILIBDIR   = $(MPIDIR)/lib
MPIINCDir   = $(MPIDIR)/include
MPILIBS     = -lmpi -lmpi_cxx -lmpi_mpifh

# for install option only
PREFIX      = $(HOME)/FrontISTR
BINDIR      = $(PREFIX)/bin
LIBDIR      = $(PREFIX)/lib
INCLUDEDIR  = $(PREFIX)/include

# Metis
METISDIR    = $(HOME)/.local
METISLIBDIR = $(METISDIR)/lib
METISINCDir = $(METISDIR)/include
HECMW_METIS_VER= 5

# ParMetis
PARMETISDIR = $(HOME)/ParMetis-3.1
PARMETISLIBDIR = $(PARMETISDIR)
PARMETISINCDir = $(PARMETISDIR)/ParMETISLib

# Refiner
REFINERDIR  = $(HOME)/Software/REVOCAP_Refiner-1.1.04
REFINERINCDir = $(REFINERDIR)/Refiner
REFINERLIBDIR = $(REFINERDIR)/lib/x86_64-linux

# Coupler
REVOCAPDIR  = $(HOME)/REVOCAP_Coupler
REVOCAPINCDir = $(REVOCAPDIR)/librcap
REVOCAPLIBDIR = $(REVOCAPDIR)/librcap
```

```
# MUMPS
MUMPSDIR    = $(HOME)/.local
MUMPSINCDir = $(MUMPSDIR)/include
MUMPSLIBDIR = $(MUMPSDIR)/lib
MUMPSLIBS   = -ldmumps -lmumps_common -lpord -L$(HOME)/.local/lib -lscalapack

# MKL PARDISO
MKLDIR      = $(HOME)/
MKLINCDIR   = $(MKLDIR)/include
MKLLIBDIR   = $(MKLDIR)/lib

# ML
MLDIR       = $(HOME)/.local
MLINCDir    = $(MLDIR)/include
MLLIBDIR    = $(MLDIR)/lib
MLLIBS      = -lml -lamesos -ltrilinoss -lzoltan -lepetra -lteuchosremainder -
lteuchosnumerics -lteuchoscomm -lteuchosparameterlist -lteuchoscore -lepetra

# C compiler settings
CC          = mpicc -fopenmp
CFLAGS      =
LDFLAGS     = -lstdc++ -lm
OPTFLAGS    = -O3

# C++ compiler settings
CPP         = mpicxx -fopenmp
CPPFLAGS    =
CPPLDFlags  =
CPPOPTFlags = -O3

# Fortran compiler settings
F90         = mpif90 -fopenmp
F90FLAGS    =
F90LDFLAGS  = -lstdc++ -L$(HOME)/.local/lib -lopenblas
F90OPTFlags = -O2
F90FPP      = -cpp
F90LINKER   = mpif90 -fopenmp

MAKE        = make
AR          = ar -ruv
MV          = mv -f
CP          = cp -f
RM          = rm -f
MKDIR       = mkdir -p
```

実行

OpenMP並列(スレッド並列、共有メモリ)

```
$ export OMP_NUM_THREADS=4
```

```
$ fistr1
```

```
▼ Terminal - michioga@michioga-VirtualBox: ~/Software/fistr/FrontISTR/tutorial/01_el - + ×
ファイル(F) 編集(E) 表示(V) ターミナル(T) タブ(A) ヘルプ(H)

166 2.120814E-08
167 1.629688E-08
168 1.258699E-08
169 9.70012E-09
### Relative residual = 9.71404E-09

### summary of linear solver
169 iterations 9.714037E-09
set-up time : 5.656880E+00
solver time : 1.169496E+01
solver/comm time : 2.976146E-02
solver/matvec : 5.742754E+00
solver/precond : 1.014706E+01
solver/1 iter : 6.920092E-02
work ratio (%) : 9.974552E+01

Start visualize PSF 1 at timestep 1
### FSTR_SOLVE_NLGEOM FINISHED!

=====
TOTAL TIME (sec) : 20.38
pre (sec) : 0.56
solve (sec) : 19.82
=====
```

MPI並列 (プロセス並列、分散メモリ)

```
$ export OMP_NUM_THREADS=1
```

```
$ hecmw_part1
```

```
$ mpirun -np 4 fistr1
```

```
▼ Terminal - michioga@michioga-VirtualBox: ~/Software/fistr/FrontISTR/tutorial/02_el - + ×
ファイル(F) 編集(E) 表示(V) ターミナル(T) タブ(A) ヘルプ(H)

102 2.077044E-08
103 1.565934E-08
104 1.161198E-08
105 8.549157E-09
### Relative residual = 8.57098E-09

### summary of linear solver
105 iterations 8.570977E-09
set-up time : 1.912429E+00
solver time : 1.138994E+01
solver/comm time : 7.718301E-01
solver/matvec : 6.422610E+00
solver/precond : 9.709771E+00
solver/1 iter : 1.084756E-01
work ratio (%) : 9.322358E+01

Start visualize PSF 1 at timestep 1
### FSTR_SOLVE_NLGEOM FINISHED!

=====
TOTAL TIME (sec) : 15.07
pre (sec) : 0.21
solve (sec) : 14.86
=====
```

コツやエラー対処など

コンパイルが出来ないときチェックしてみてください

外部ライブラリ構築のコツ

OpenMPを有効化する場合、全てのライブラリもOpenMP化しておく。

- OpenBLAS (make USE_OPENMP=1)
- METIS (make config openmp=1)
- ScaLAPACK (cmake -DCMAKE_EXE_LINKER_FLAGS="-fopenmp") ※ライブラリそのものは関係無し
- MUMPS (Makefile.incの中で -fopenmp を指定)
- Trilinos ML (cmake -DTrilinos_ENABLE_OpenMP=ON)

各ライブラリのインストール先は \$HOME/.local 以下が良い

- \$HOME/.local/bin にパスを通しておくと良い
 - CentOS7.3の場合既にPATHが通っている
 - cmake でライブラリやヘッダファイルの探索が可能になる
- make install が用意されていないものは、\$HOME/.local/lib にライブラリ、\$HOME/.local/include にヘッダファイルをコピーすると良い

ビルド時のエラーの対処

- 致命的エラー: `xx.h`: そのようなファイルやディレクトリがありません
 - `Makefile.conf`中の `XXXINCDIR =` の指定が間違っているかもしれません
- `/usr/bin/ld: -lxxx` が見つかりません
 - `Makefile.conf`中の `XXXLIBDIR=` の指定が間違っているかもしれません
- `xxx.cpp`: `(.text._ZXXX) 'operator delete(void*)'` に対する定義されていない参照です
 - C++のリンカーの指定かC++で必要とするライブラリの指定が間違っているかもしれません(`-lstdc++`など)。
- エラー: `unrecognized command line option '-fpp'`
 - `Makefile.conf`中の `F90FPP =` の指定を `'-cpp'` に変更してください。
- `'GOMP_xxx_xxx'` に対する定義されていない参照です
 - `'-fopenmp'` を指定してされていないかもしれません。
- `'dgemm_'` に対する定義されていない参照です
 - LAPACK(OpenBLAS)の参照先が間違っているかもしれません。
- 関数 `'MPI::XXX::XXX'` に対する定義されていない参照です
 - C++用MPIライブラリ(`-lmpi_cxx`)の指定が間違っているかもしれません。
- `'blacs_xxx_'` に対する定義されていない参照です
 - `-lscalapack` が指定されていない可能性があります。
- `-lscalapack`を指定したのにも関わらず、`'daxpy_'`に対する定義されていない参照です が出る
 - `-lscalapack` と `-lopenblas` の順序が逆の可能性があります。
- エラー: Function `'metis_mcpartgraphrecursive'` at (1) has no IMPLICIT type
 - `Makefile.conf`中の `HECMW_METIS_VER=`の値が間違っているかもしれません。

実行時のエラー

- コンパイルは通ったが、実行できない
- MPIの共有ライブラリが見えていないのかもしれません。CentOS7.3では以下のように見えます。

```
$ ldd fistr1
linux-vdso.so.1 => (0x00007fff58b76000)
libmpi.so.12 => /usr/lib64/openmpi/lib/libmpi.so.12 (0x00007efddfd4000)
libmpi_cxx.so.1 => /usr/lib64/openmpi/lib/libmpi_cxx.so.1 (0x00007efddfbbe000)
libstdc++.so.6 => /lib64/libstdc++.so.6 (0x00007efddf897000)
libmpi_usempi.so.5 => /usr/lib64/openmpi/lib/libmpi_usempi.so.5 (0x00007efddf694000)
libmpi_mpifh.so.12 => /usr/lib64/openmpi/lib/libmpi_mpifh.so.12 (0x00007efddf43e000)
libgfortran.so.3 => /lib64/libgfortran.so.3 (0x00007efddf11b000)
libm.so.6 => /lib64/libm.so.6 (0x00007efdde19000)
libgomp.so.1 => /lib64/libgomp.so.1 (0x00007efddebf2000)
libgcc_s.so.1 => /lib64/libgcc_s.so.1 (0x00007efdde9dc000)
libquadmath.so.0 => /lib64/libquadmath.so.0 (0x00007efdde7a0000)
libpthread.so.0 => /lib64/libpthread.so.0 (0x00007efdde583000)
libc.so.6 => /lib64/libc.so.6 (0x00007efdde1c2000)
/lib64/ld-linux-x86-64.so.2 (0x00007efde00bf000)
libopen-rte.so.12 => /usr/lib64/openmpi/lib/libopen-rte.so.12 (0x00007efddd45000)
libopen-pal.so.13 => /usr/lib64/openmpi/lib/libopen-pal.so.13 (0x00007efdddca0000)
libdl.so.2 => /lib64/libdl.so.2 (0x00007efddd9c000)
librt.so.1 => /lib64/librt.so.1 (0x00007efddd894000)
libutil.so.1 => /lib64/libutil.so.1 (0x00007efddd690000)
libhwloc.so.5 => /lib64/libhwloc.so.5 (0x00007efddd456000)
libnuma.so.1 => /lib64/libnuma.so.1 (0x00007efddd249000)
libltdl.so.7 => /lib64/libltdl.so.7 (0x00007efddd03f000)
```


Intelコンパイラを利用する場合(1)

Intelコンパイラを利用する場合

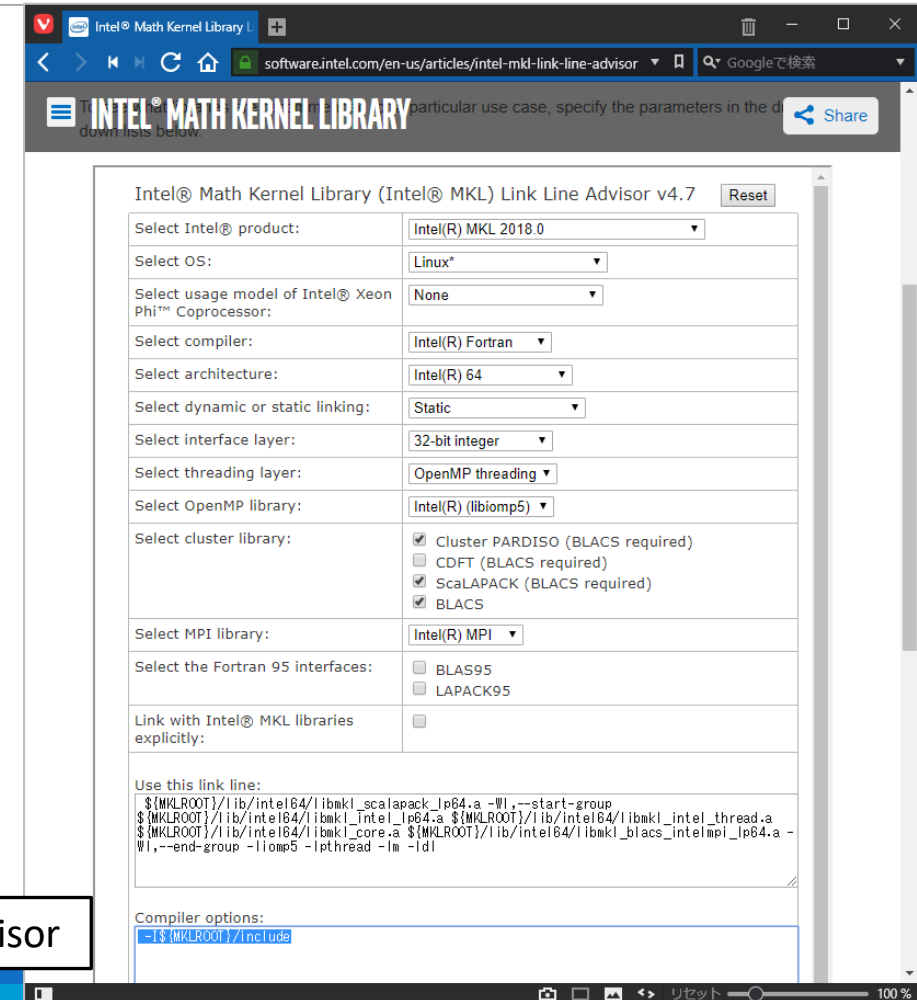
- コンパイラを gcc/gfortran から icc/ifort へ変更
- OpenMPIの代わりに、Intel MPIを使用
- OpenBLAS・ScaLAPACKの代わりに、Intel MKLを使用
- OpenMPのコンパイルフラグ "-fopenmp" を "-openmp" に変更
- プリプロセッサ起動のフラグを "-cpp" から "-fpp" へ変更

Intelコンパイラを利用する場合(2)

ライブラリやオプションを調べるには Intel Math Kernel Library Link Line Advisorが便利です

- ✓ Select OS : Linux
- ✓ Select usage model of Intel Xeon Phi Coprocessor : None
- ✓ Select compiler : Intel(R) Fortran
- ✓ Select architecture : Intel(R) 64
- ✓ Select dynamic or static linking : 好みで
- ✓ Select interface layer : 32-bit integer
- ✓ Select threading layer : OpenMP threading
- ✓ Select OpenMP library : Intel(R) (libiomp5)
- ✓ Select cluster library : Cluster PARDISO, ScaLAPACK, BLACS
- ✓ Select MPI library : Intel(R) MPI

<https://software.intel.com/en-us/articles/intel-mkl-link-line-advisor>



Intel® Math Kernel Library (Intel® MKL) Link Line Advisor v4.7

Select Intel® product: Intel(R) MKL 2018.0

Select OS: Linux*

Select usage model of Intel® Xeon Phi™ Coprocessor: None

Select compiler: Intel(R) Fortran

Select architecture: Intel(R) 64

Select dynamic or static linking: Static

Select interface layer: 32-bit integer

Select threading layer: OpenMP threading

Select OpenMP library: Intel(R) (libiomp5)

Select cluster library:
☒ Cluster PARDISO (BLACS required)
☐ CDFT (BLACS required)
☒ ScaLAPACK (BLACS required)
☒ BLACS

Select MPI library: Intel(R) MPI

Select the Fortran 95 interfaces:
☐ BLAS95
☐ LAPACK95

Link with Intel® MKL libraries explicitly: ☐

Use this link line:
\$ {MKLROOT}/lib/intel64/libmkl_scalapack_lp64.a -Wl,--start-group
\$ {MKLROOT}/lib/intel64/libmkl_intel_lp64.a \$ {MKLROOT}/lib/intel64/libmkl_intel_thread.a
\$ {MKLROOT}/lib/intel64/libmkl_core.a \$ {MKLROOT}/lib/intel64/libmkl_blacs_intelmpi_lp64.a
-Wl,--end-group -liomp5 -lpthread -lm -ldl

Compiler options:
-I \$ {MKLROOT}/include

FrontISTR V5.0(開発中)で導入される 新しいビルド方法の紹介

次期バージョン FrontISTR V5.0では、cmakeを使った新しいビルド方法が導入される予定です。

FrontISTR V5.0(開発中)のビルド方法

FrontISTR V5.0から

- Makefile.conf & setup.sh でのビルド方法に加え
 - CMakeでのビルド方法
- がサポートされる予定です。

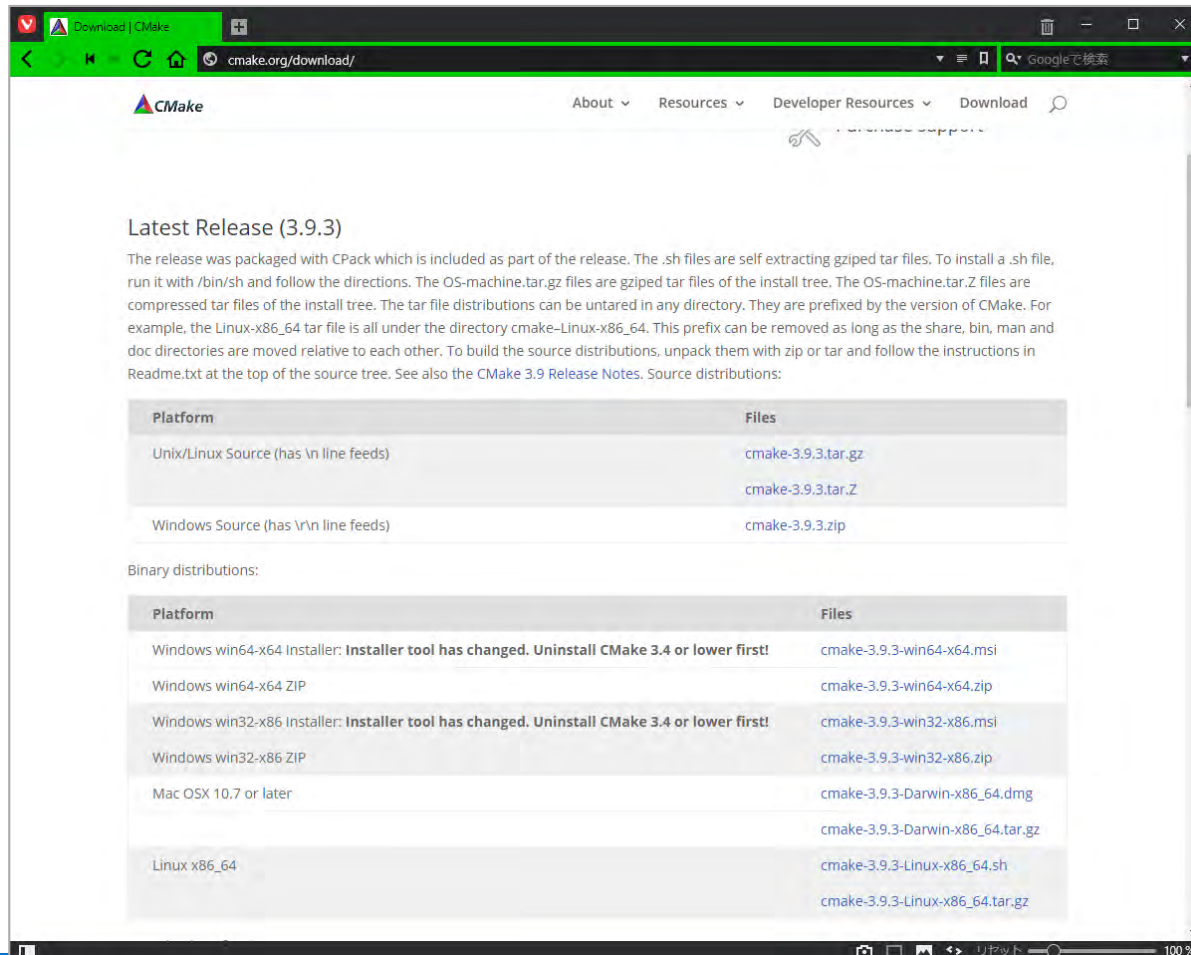
※すでに外部ライブラリの metis-5.1.0 や scalapack-2.0.2 でもビルド方法として採用されています。

※ 外部ライブラリの作成は必要です。

CMakeとは

- ビルドするプラットフォームやコンパイルに極力依存しないようなメタビルドシステム
- コンパイラによる、フラグの違いやライブラリ・ヘッダファイルの場所をほぼ自動で補完
- ソースコード間の依存関係も考慮したMakefileを生成
 - make コマンドは依然として必要
- cmakeは、多くのディストリビューションで配布されている。
- 公式サイトでは、最新版のバイナリも配布されている。
- cmakeそのものは C++で記述されており、他のライブラリとの依存は極力排除されているためソースからのコンパイルも容易。
- 並列コンパイルが可能となり、ビルド時間の短縮が期待できる。
- CentOS7.3、Ubuntu16.04、MacOS X、Windows10(mingw64)、京、FX10で動作報告あり。

CMakeの導入方法



最新版は

<https://cmake.org/download/>
からバイナリをダウンロード

- win64-x64
- win32-x86
- Mac OSX 10.7 or later
- Linux x86_64

ソースから

```
$ tar xvf cmake-3.9.3.tar.gz  
$ cd cmake-3.9.3  
$ ./bootstrap  
$ make
```

CentOS7.3のパッケージから

```
# yum install cmake
```

Ubuntu16.04のパッケージから

```
# apt install cmake
```

FrontISTR V5.0(開発中) を CMakeでビルド

FrontISTR V5.0 (開発中)

```
$ tar xvf FrontISTR_V50.tar.gz  
$ cd FrontISTR_V50  
$ mkdir build; cd build  
$ cmake ..  
$ make -j4  
$ make install
```

FrontISTR V4.6以前

```
$ tar xvf FrontISTR_V46.tar.gz  
$ cd FrontISTR_V46  
$ cp Makefile.conf.org Makefile.conf  
$ vi Makefile.conf  
$ ./setup.sh -p --with-tools --with-refiner --with-lapack ¥  
--with-metis --with-mumps --with-mt  
$ make  
$ make install
```

ビルド時間 : 42秒

ビルド時間 : 2分00秒

CMake実行時画面

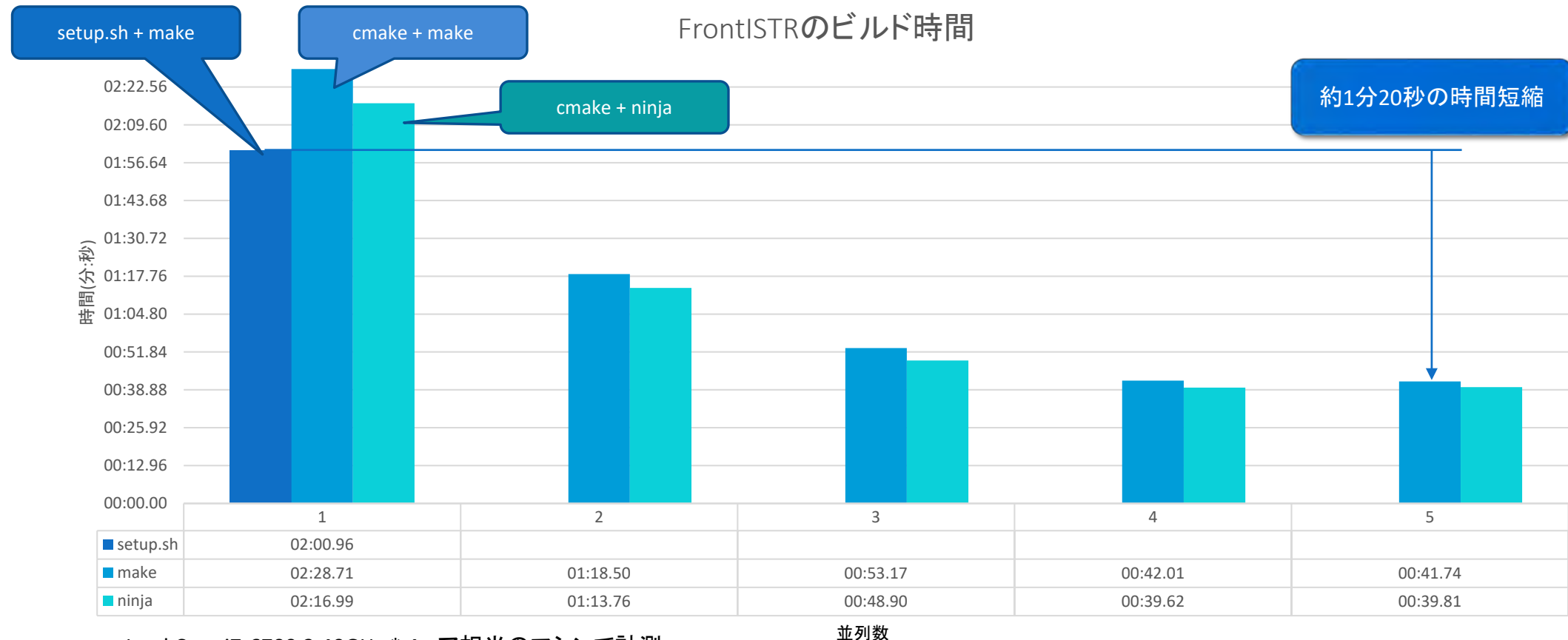
```
Terminal - michioga@localhost:~/Software/FrontISTR/build
ファイル(F) 編集(E) 表示(V) ターミナル(T) タブ(A) ヘルプ(H)
-- Performing Test OpenMP_FLAG_DETECTED - Success
-- Try OpenMP CXX flag = [-fopenmp]
-- Performing Test OpenMP_FLAG_DETECTED - Success
-- Found OpenMP: -fopenmp
-- Looking for include file pthread.h - found
-- Looking for pthread_create in pthread - not found
-- Looking for pthread_create in pthreads - not found
-- Looking for pthread_create in pthread - found
-- Found Threads: TRUE
-- A library with BLAS API found.
-- A library with LAPACK API found.
-- Found Metis: /home/michioga/Software/metis-5.1.0/build/Linux-x86_64/libmetis/
libmetis.a
-- Found MUMPS: /home/michioga/Software/MUMPS_5.1.1/lib/libdmumps.a;/home/michioga/
Software/MUMPS_5.1.1/lib/libdmumps_common.a;/home/michioga/Software/MUMPS_5.1.
1/lib/libdpor.a
-- Could NOT find Parmetis (missing: PARMETIS_LIBRARIES PARMETIS_INCLUDE_PATH)
-- Found REVOCAP Refiner: /home/michioga/Software/REVOCAP_Refiner-1.1.04/lib/x86
_64-linux/libRcapRefiner.a
-- Could NOT find REVOCAP_Coupler (missing: REVOCAP_LIBRARIES REVOCAP_INCLUDE_P
ATH)
-- Found Scalapack: /home/michioga/Software/scalapack-2.0.2/build/lib/libscalapa
ck.a
-- Found Doxygen: /usr/bin/doxygen (found version "1.8.5")
-- The following OPTIONAL packages have been found:
* MPI
* OpenMP
* BLAS
* Threads
* LAPACK
* Metis
* Mumps
* Refiner
* Scalapack
* Trilinos
* Doxygen
-- The following OPTIONAL packages have not been found:
* Parmetis
* Revocap
-- Build FrontISTR for Linux-x86_64-3.10.0-693.2.2.el7.x86_64
-- BUILD Type : RELEASE
-- C compiler : /usr/bin/cc
-- CXX compiler : /usr/bin/c++
-- Fortran compiler : /usr/bin/f95
-- Installation path : /home/michioga/local
-- Found Ruby: /usr/bin/ruby (found version "2.0.0")
-- Found /usr/bin/ruby. Enable tests.
If you would like to test, please type 'make test' after build binaries.
-- Configuring done
-- Generating done
-- Build files have been written to: /home/michioga/Software/FrontISTR/build
[michioga@localhost build]$
```

cmake中の画面

```
Terminal - michioga@localhost:~/Software/FrontISTR/build
ファイル(F) 編集(E) 表示(V) ターミナル(T) タブ(A) ヘルプ(H)
Building C object hecmw/tools/CMakeFiles/hec2rcap.dir/hec2rcap/hec2rcap.c.o
[ 51%] Building C object hecmw/tools/CMakeFiles/hecmw_part1.dir/partitioner/hec
mw_part1ex.c.o
Linking CXX executable hecmw_vis1
Linking CXX executable rconv
Linking CXX executable hec2rcap
[ 51%] Built target rconv
Scanning dependencies of target rmerge
[ 51%] Building C object hecmw/tools/CMakeFiles/rmerge.dir/result_file_merger/f
ile_rmerge_util.c.o
[ 51%] [ 51%] Built target hecmw_vis1
Building C object hecmw/tools/CMakeFiles/hecmw_part3.dir/partitioner/hecmw_part
3_lag.c.o
[ 51%] Built target hec2rcap
[ 51%] Building C object hecmw/tools/CMakeFiles/hecmw_part1.dir/partitioner/hec
mw_mesh_hash_sort.c.o
Scanning dependencies of target fistr
[ 51%] Building C object hecmw/tools/CMakeFiles/hecmw_part1.dir/partitioner/hec
mw_mesh_edge_info.c.o
[ 51%] Building Fortran object fistr/CMakeFiles/fistr.dir/src/libelement/quadr
ature.f90.o
[ 52%] Building C object hecmw/tools/CMakeFiles/rmerge.dir/result_file_merger/f
ile_rmerge.c.o
[ 52%] Building Fortran object fistr/CMakeFiles/fistr.dir/src/libelement/hecmw
```

並列make中の画面

ビルド時間の比較



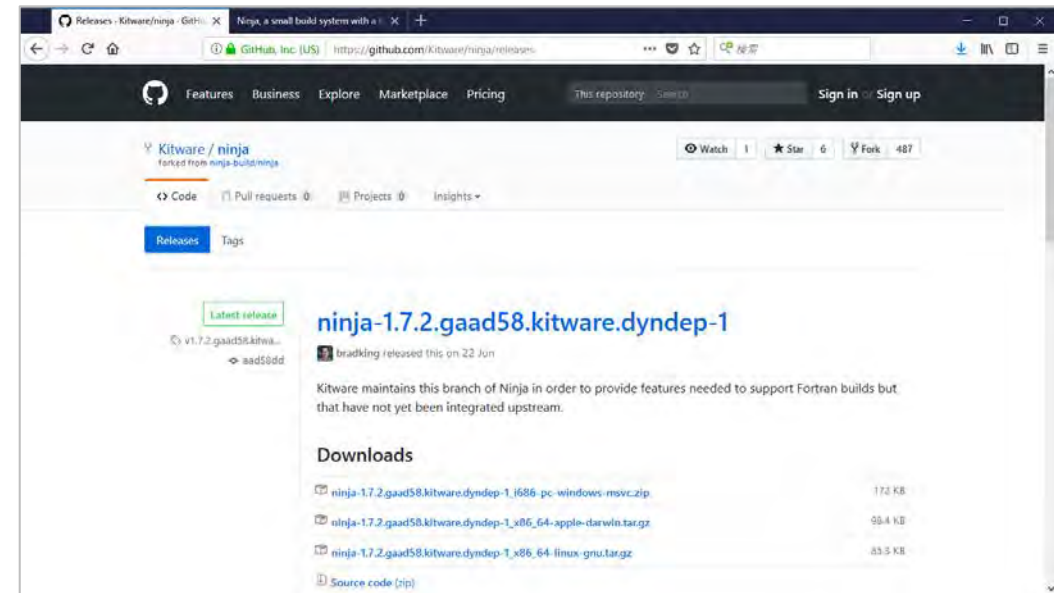
Intel Core i7-6700 3.40GHz * 4コア相当のマシンで計測
time コマンドで3回の平均値

並列数

ninja (Fortran対応版) : <https://github.com/Kitware/ninja/releases>

Ninja ビルドツール

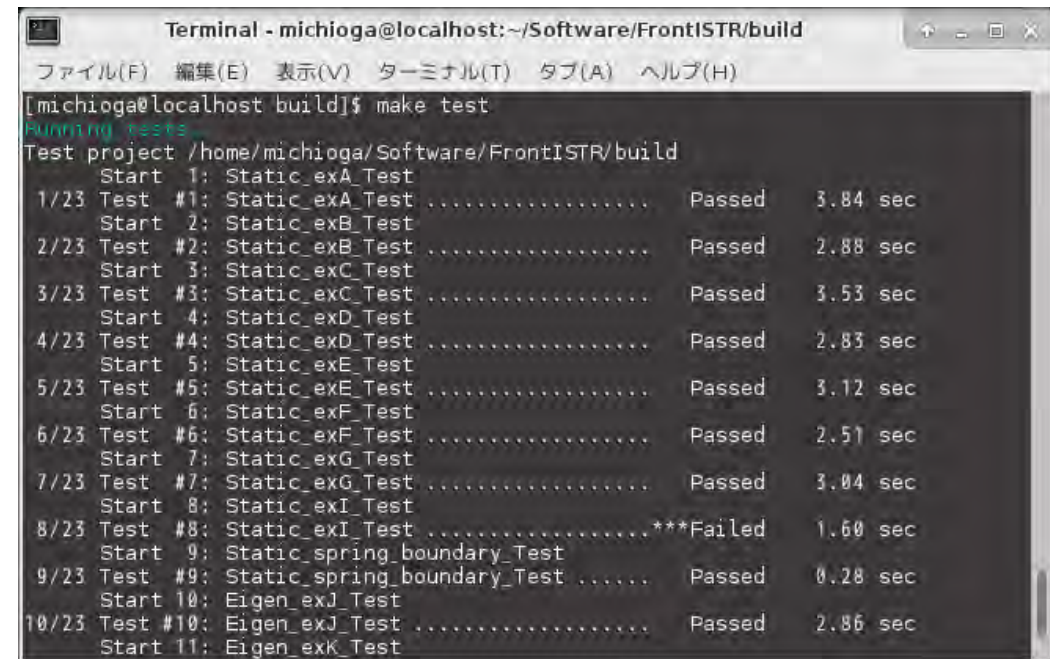
- make の置き換えを目的に開発されたツール
- 高速性に重点を置いて開発されている
- 本家(<https://ninja-build.org/>)版は、Fortran に対応していない
- Kitware版
(<https://github.com/Kitware/ninja/releases>)を使うことで、Fortranもコンパイル出来るようになる。
- makeに比べると、例では最大12秒速かった。



簡易テスト機能

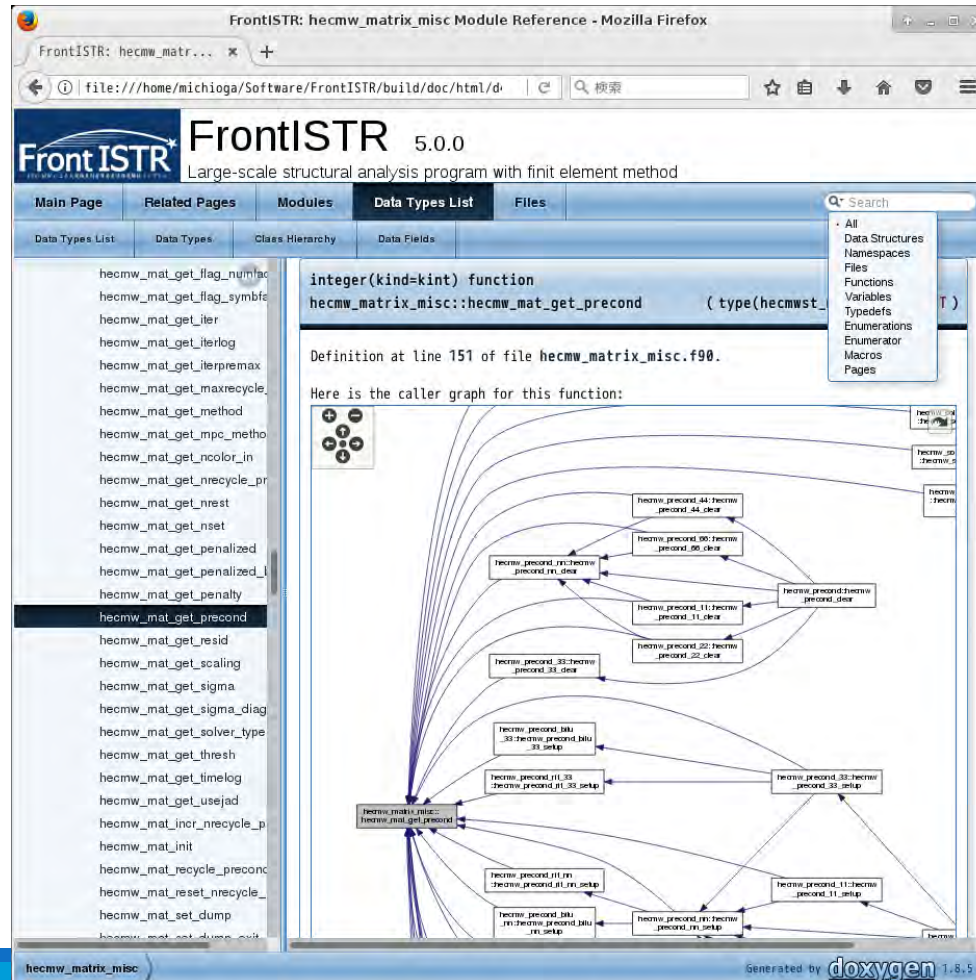
CMakeに簡易テスト機能が統合されました

- make test による簡易テストが出来るようになりました。
- テストにはrubyが必要になります。
- examplesディレクトリの例題を実行し、正解結果ファイルと突き合わせて、違いがあれば表示されます。



```
Terminal - michioga@localhost:~/Software/FrontISTR/build
ファイル(F) 編集(E) 表示(V) ターミナル(T) タブ(A) ヘルプ(H)
[michioga@localhost build]$ make test
Running tests...
Test project /home/michioga/Software/FrontISTR/build
  Start 1: Static_exA_Test
1/23 Test #1: Static_exA_Test ..... Passed    3.84 sec
  Start 2: Static_exB_Test
2/23 Test #2: Static_exB_Test ..... Passed    2.88 sec
  Start 3: Static_exC_Test
3/23 Test #3: Static_exC_Test ..... Passed    3.53 sec
  Start 4: Static_exD_Test
4/23 Test #4: Static_exD_Test ..... Passed    2.83 sec
  Start 5: Static_exE_Test
5/23 Test #5: Static_exE_Test ..... Passed    3.12 sec
  Start 6: Static_exF_Test
6/23 Test #6: Static_exF_Test ..... Passed    2.51 sec
  Start 7: Static_exG_Test
7/23 Test #7: Static_exG_Test ..... Passed    3.04 sec
  Start 8: Static_exI_Test
8/23 Test #8: Static_exI_Test .....***Failed  1.60 sec
  Start 9: Static_spring_boundary_Test
9/23 Test #9: Static_spring_boundary_Test ..... Passed    0.28 sec
  Start 10: Eigen_exJ_Test
10/23 Test #10: Eigen_exJ_Test ..... Passed    2.86 sec
  Start 11: Eigen_exK_Test
```

ソースコードのドキュメンテーション



CMakeにソースコードのドキュメンテーションが統合されました。

- doxygen および graphviz が必要になります。
- cmake -DWITH_DOC=ON .. を指定します。
- ソースコードに埋め込まれたコメントや関数の依存関係がHTMLに変換されます。
- 変数や関数の検索も出来るようになります。

バイナリパッケージの作成

- ▶ バイナリパッケージの作成が簡単になりました。
- ▶ CMakeに同梱されている cpack を使い
 - ▶ deb, rpm, tar.bz2 のパッケージを作成できます。

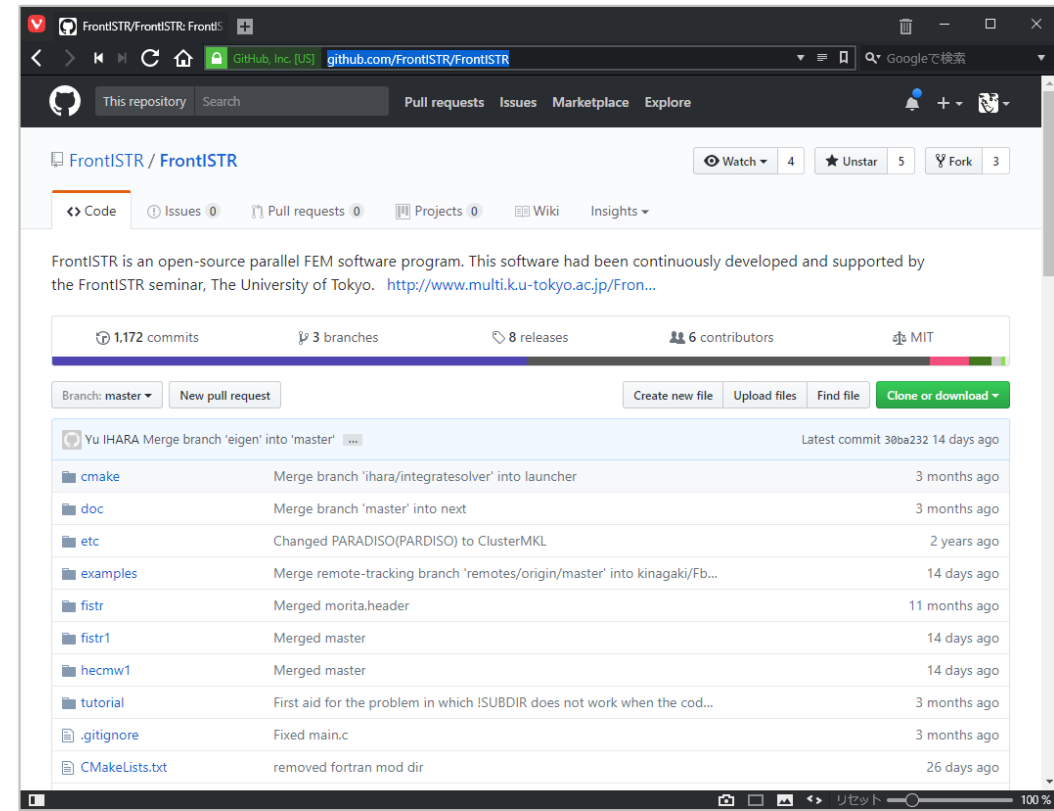
```
Terminal - michioga@localhost:~/Software/FrontISTR/build
ファイル(F) 編集(E) 表示(V) ターミナル(T) タブ(A) ヘルプ(H)
CPack: Create package using DEB
CPack: Install projects
CPack: - Run preinstall target for: FrontISTR
CPack: - Install project: FrontISTR
CPack: Create package
CPack: - package: /home/michioga/Software/FrontISTR/build/FrontISTR-5.0.0-Linux.
deb generated.
CPack: Create package using RPM
CPack: Install projects
CPack: - Run preinstall target for: FrontISTR
CPack: - Install project: FrontISTR
CPack: Create package
CPackRPM: Will use GENERATED spec file: /home/michioga/Software/FrontISTR/build/
_CPack_Packages/Linux/RPM/SPECS/frontistr.spec
CPack: - package: /home/michioga/Software/FrontISTR/build/FrontISTR-5.0.0-Linux.
rpm generated.
[michioga@localhost build]$ ls
CMakeCache.txt          FrontISTR-5.0.0-Linux.rpm      cmake_install.cmake
CMakeFiles              FrontISTR-5.0.0-Linux.tar.bz2 doc
CPackConfig.cmake       FrontISTRConfig.h             examples
CPackSourceConfig.cmake Makefile                       fletri
CTestTestfile.cmake     Testing                        hecmwl
FrontISTR-5.0.0-Linux.deb CPack_Packages                 install_manifest.txt
[michioga@localhost build]$
```

CMake導入によるメリット (まとめ)

1. ビルド手順が簡単になった(Windowsでのビルドも含め)。
2. 並列コンパイルにより、ビルドの時間が短縮された。
3. ビルド後の簡易テストが手元で出来るようになった。
4. ソースコードのドキュメンテーション化により、プログラム内部の見通しが改善された。
5. バイナリパッケージの作成が容易になった。

FrontISTR V5.0(開発中) github.com

- <https://github.com/FrontISTR/FrontISTR> で FrontISTR V5.0 の開発版を公開しています
 - 開発中ですので、不具合があるかもしれませんがご報告いただけるとありがたいです。
- 最後に紹介した cmake でのビルドに対応しています。
- Pull Requestを出していただくことも可能です。



ご清聴ありがとうございました