帝京大学
TEIKYO

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)

帝京大学
TEIKYO

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

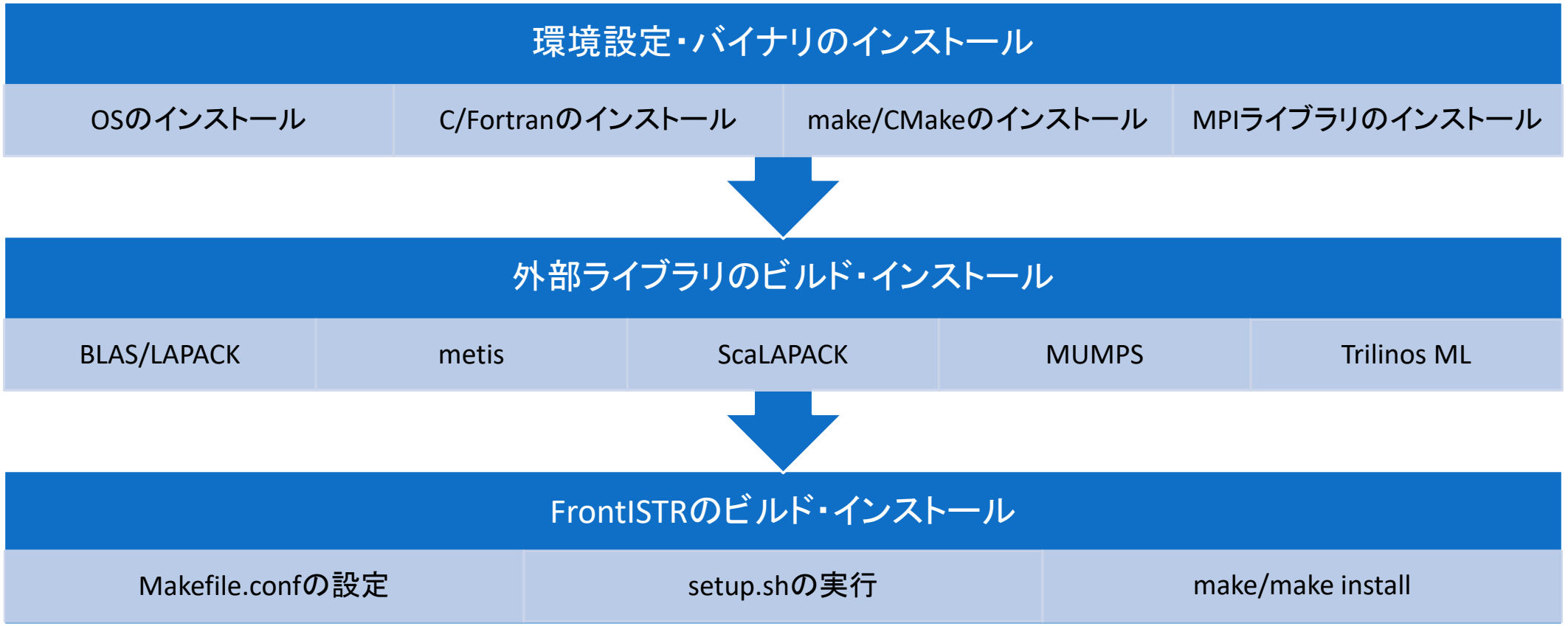
帝京大学
TEIKYO

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

帝京大学 TEIKYO

FrontISTRのビルド手順



帝京大学 TEIKYO

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。

```
$ su
# yum group mark install “Development Tools”
# yum update
```

- OpenMPIとcmakeをインストールしてください。

```
$ 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)

帝京大学 TEIKYO

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/

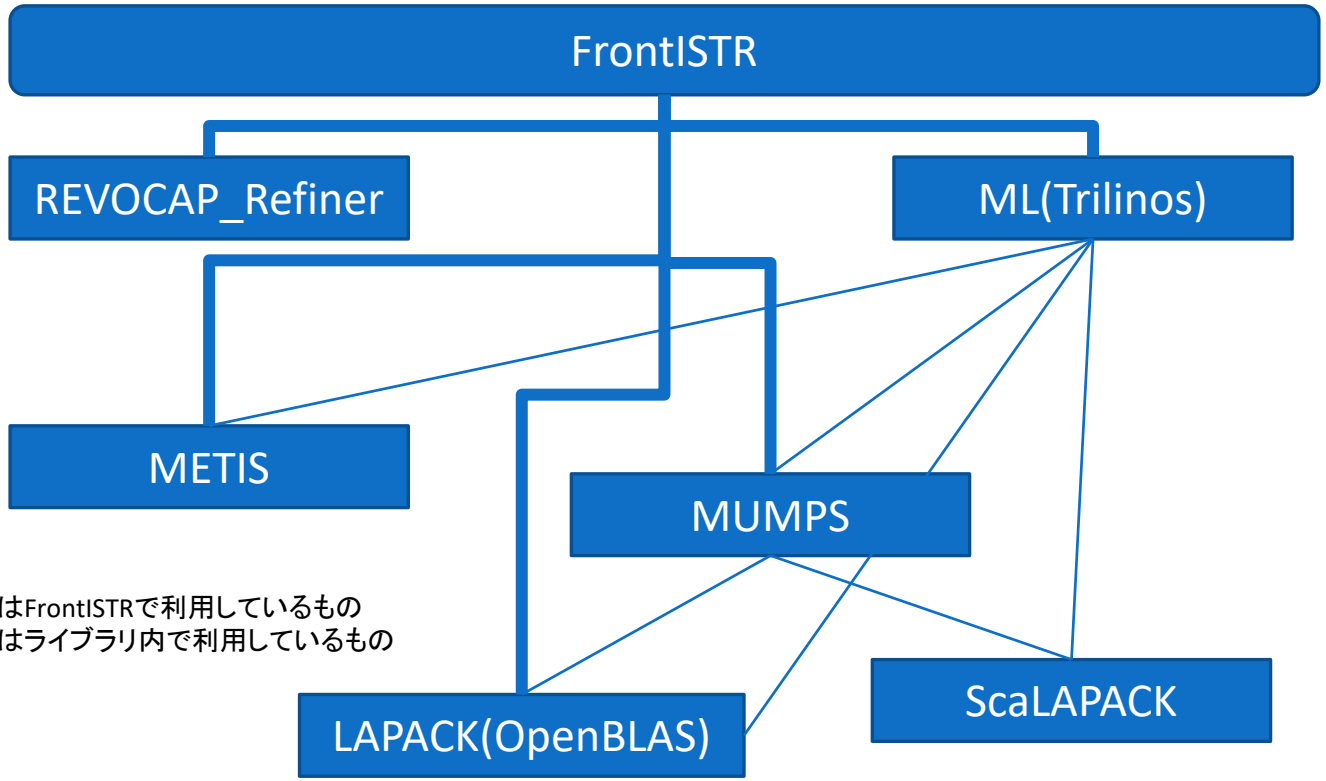
の「FrontISTRリザーバ」からダウンロードしてください。
初めての方は「ユーザID取得」から登録をしてください。

帝京大学
TEIKYO

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紹介します

ライブラリの依存関係



- 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

➢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。

帝京大学 TEIKYO

OpenBLASのビルド

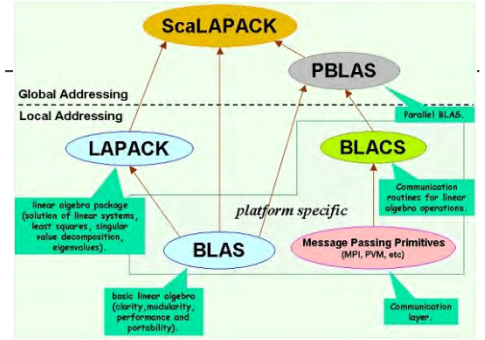
- OpenBLASは高速なフリーのBLAS/LAPACKの実装です。
- OpenMPを有効にしたライブラリを構築してください。

```
$ 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。

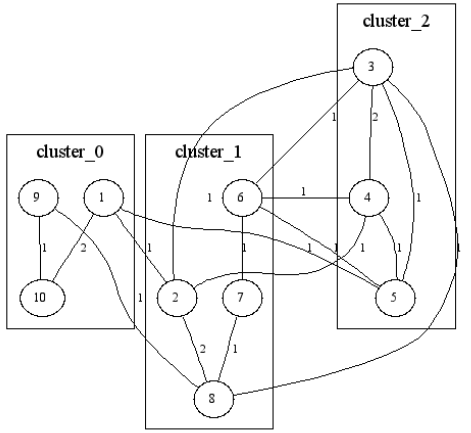
実行マシンとコンパイルマシンが違う場合、DYNAMIC_ARCH=1 を付けてください。ただし、コンパイル時間が大幅に増加します。



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。

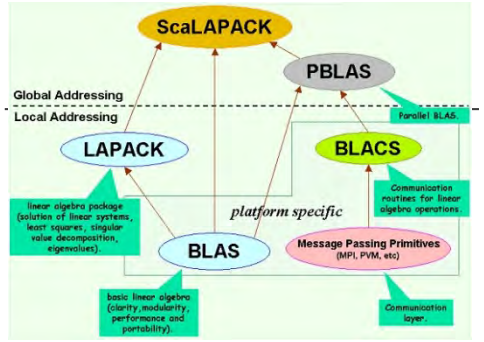
```
$ tar xvf metis-5.1.0.tar.gz
$ cd metis-5.1.0
$ make config prefix=$HOME/.local openmp=1 cc=gcc
$ make
$ make install
```



帝京大学 TEIKYO

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。

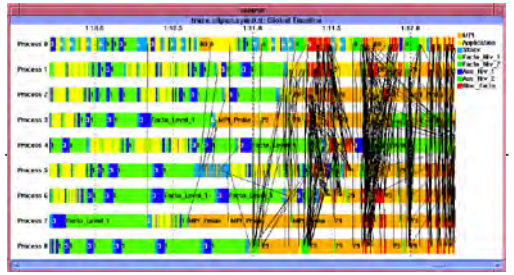



```
$ tar xvf scalapack-2.0.2.tar.gz
$ cd scalapack-2.0.2
$ mkdir build
$ cd build
$ cmake -DCMAKE_INSTALL_PREFIX=$HOME/.local -DCMAKE_EXE_LINKER_FLAGS=“-fopenmp” ¥
        -DBLAS_LIBRARIES=$HOME/.local/lib/libopenblas.a ¥
        -DLAPACK_LIBRARIES=$HOME/.local/lib/libopenblas.a
        ..
$ make
$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

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

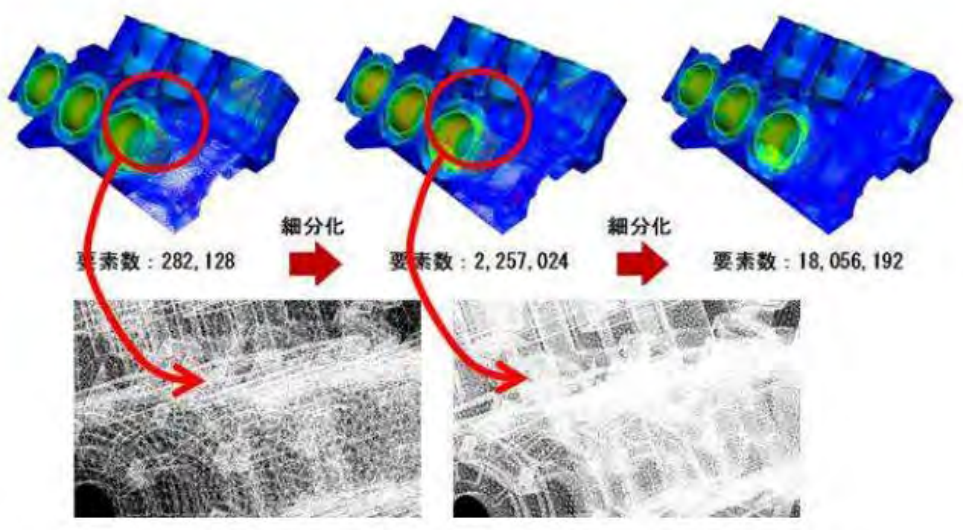
- Trilinos MLは、FrontISTRの前処理に用います。.cntファイルの中で指定が可能です。
- 構築には cmake が必要です。

```
$ 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
```



TEIKYO 帝京大学

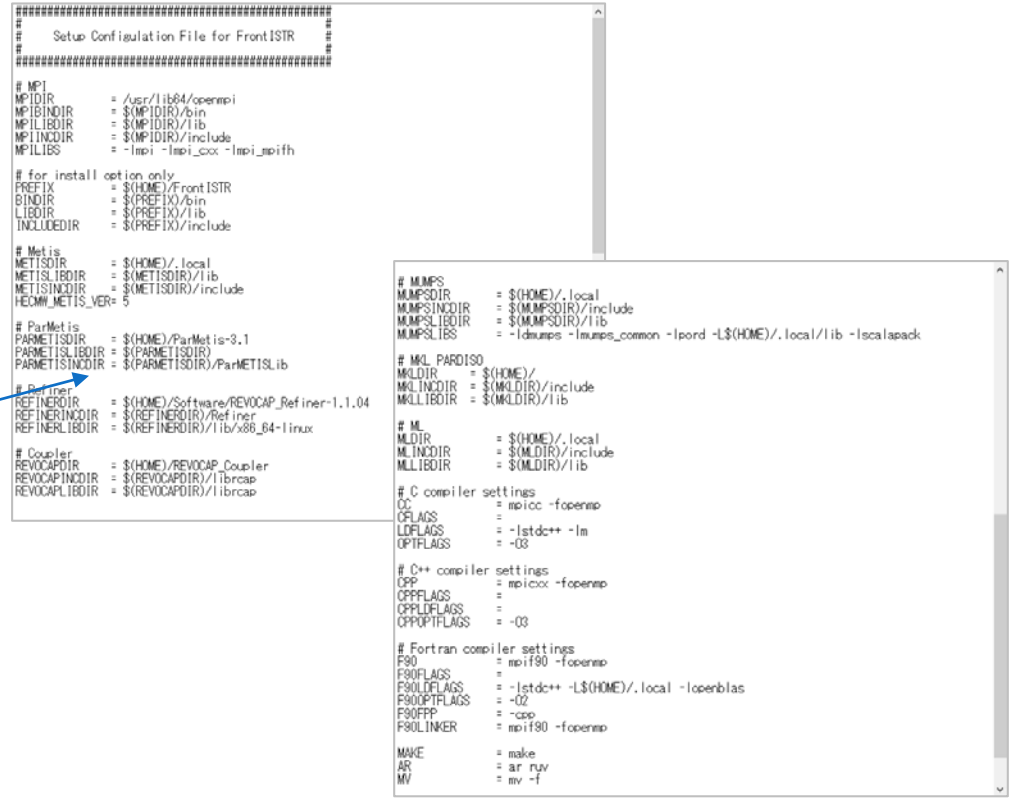
FrontISTR V4.6 のビルド

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 Configulation 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

# 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 -lopenblas
F90OPTFLAGS     = -O2
F90FPP          = -cpp
F90LINKER       = mpif90 -fopenmp

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

Makefile.conf

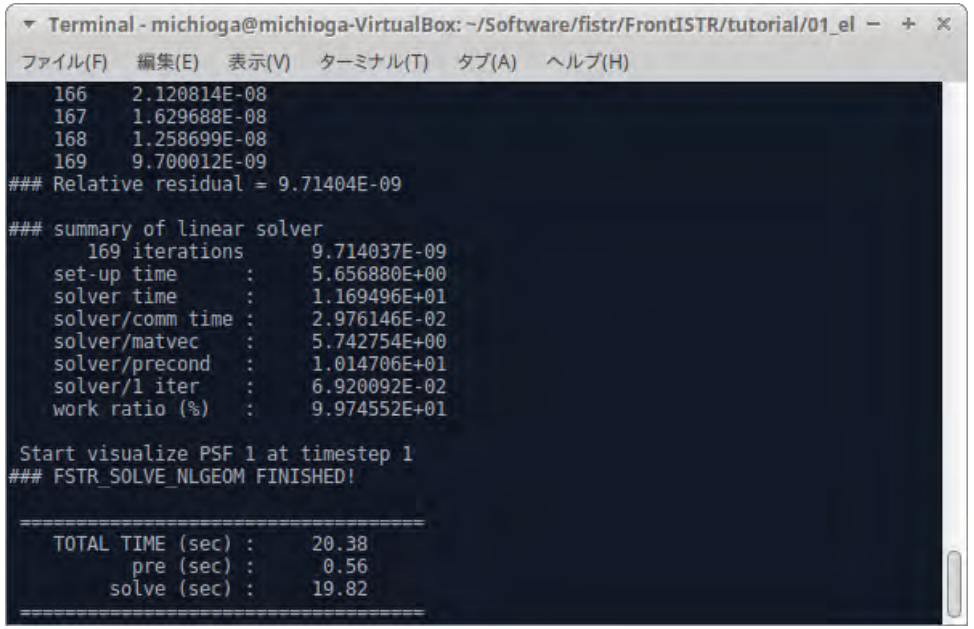
```
###################################################
#                                                 #
#     Setup Configulation File for FrontISTR      #
#                                                 #
###################################################

# 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 -ltrilinosss -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

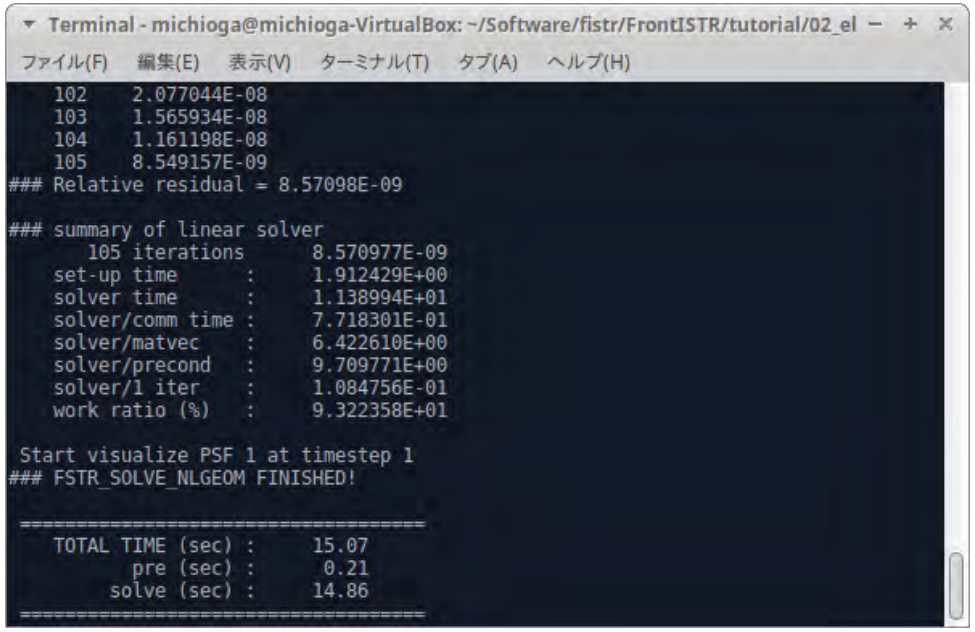
OpenMP並列(スレッド並列、共有メモリ)

$ export OMP_NUM_THREADS=4
$ fistr1



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



帝京大学
TEIKYO

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

帝京大学 TEIKYO

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 optioin '-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 (0x00007efddfdda000)
        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 (0x00007efddee19000)
        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 (0x00007efdddf45000)
        libopen-pal.so.13 => /usr/lib64/openmpi/lib/libopen-pal.so.13 (0x00007efdddca0000)
        libdl.so.2 => /lib64/libdl.so.2 (0x00007efddda9c000)
        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)
```

帝京大学 TEIKYO

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

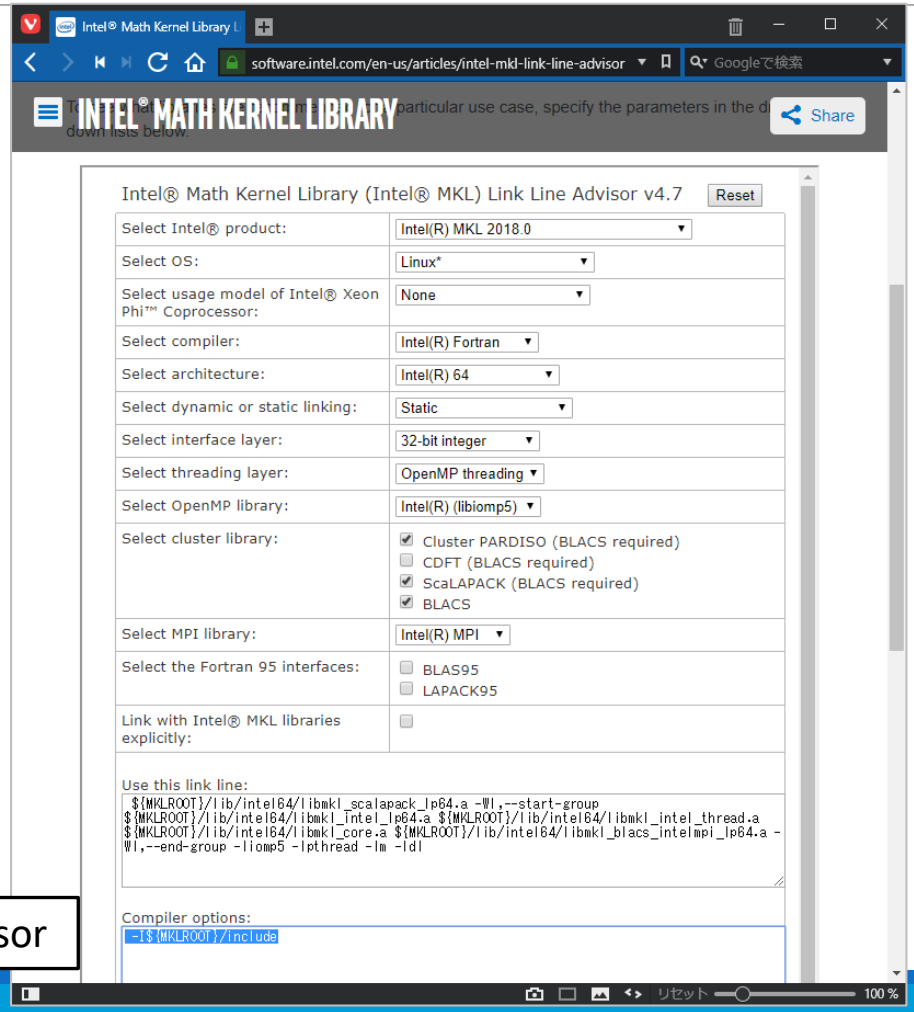
- コンパイラを gcc/gfortranからicc/ifort へ変更
- OpenMPIの代わりに、Intel MPIを使用
- OpenBLAS・ScaLAPACKの代わりに、Intel MKLを使用
- OpenMPのコンパイルフラグ "-fopenmp" を "-openmp" に変更
- プリプロセッサ起動のフラグを "-cpp" から "-fpp" へ変更

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



帝京大学
TEIKYO

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。

帝京大学 TEIKYO

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。

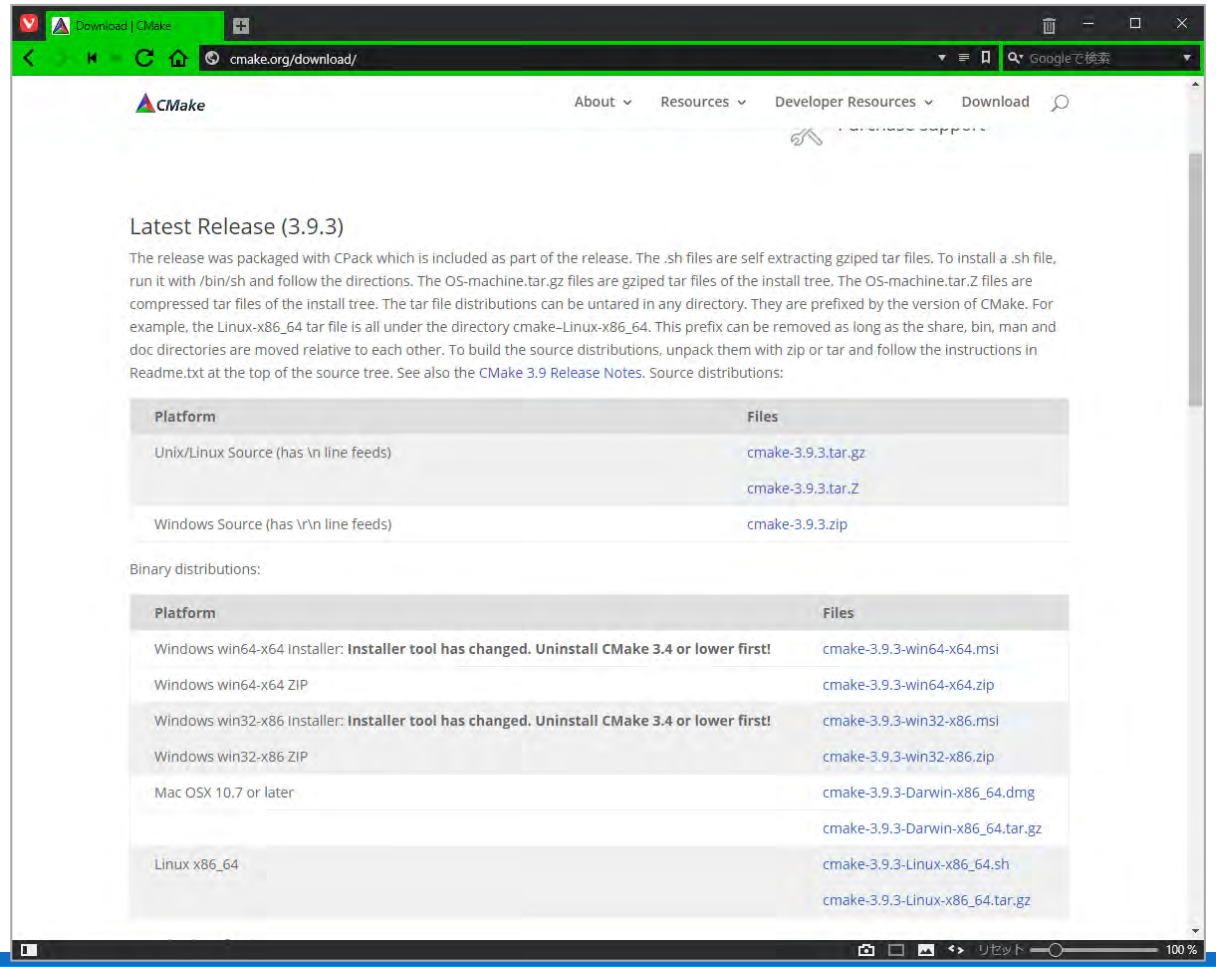
※すでに外部ライブラリの metis-5.1.0 や scalapack-2.0.2 でもビルド方法として採用されています。
※ 外部ライブラリの作成は必要です。

帝京大学 TEIKYO

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

帝京大学 TEIKYO

FrontISTR V5.0 (開発中)

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

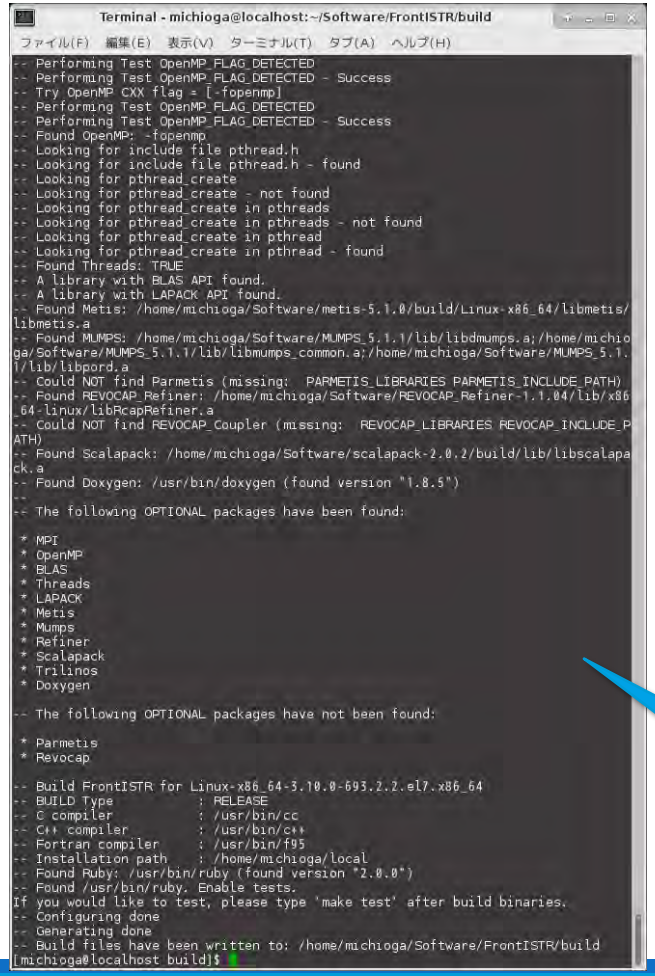
ビルド時間 : 42秒

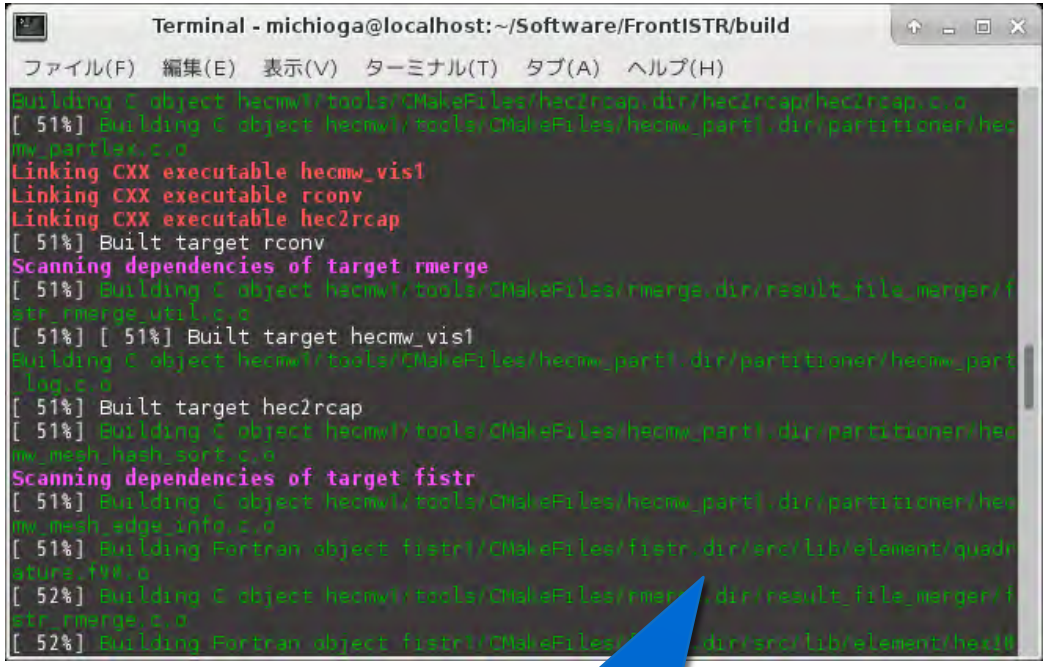
FrontISTR V4.6以前

```
$ 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-ml
$ make
$ make install
```

ビルド時間 : 2分00秒

CMake実行時画面

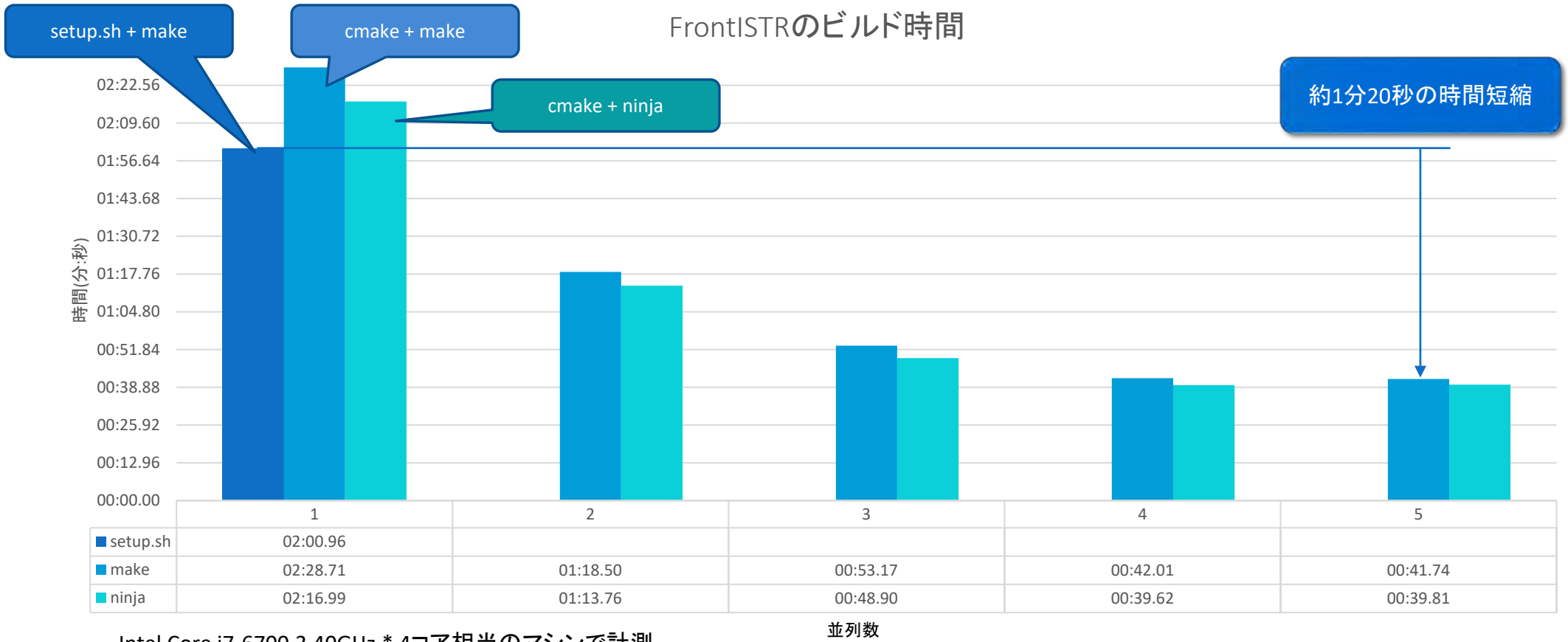




cmake中の画面

並列make中の画面

ビルド時間の比較



	1	2	3	4	5
setup.sh	02:00.96				
make	02:28.71	01:18.50	00:53.17	00:42.01	00:41.74
ninja	02:16.99	01:13.76	00:48.90	00:39.62	00:39.81

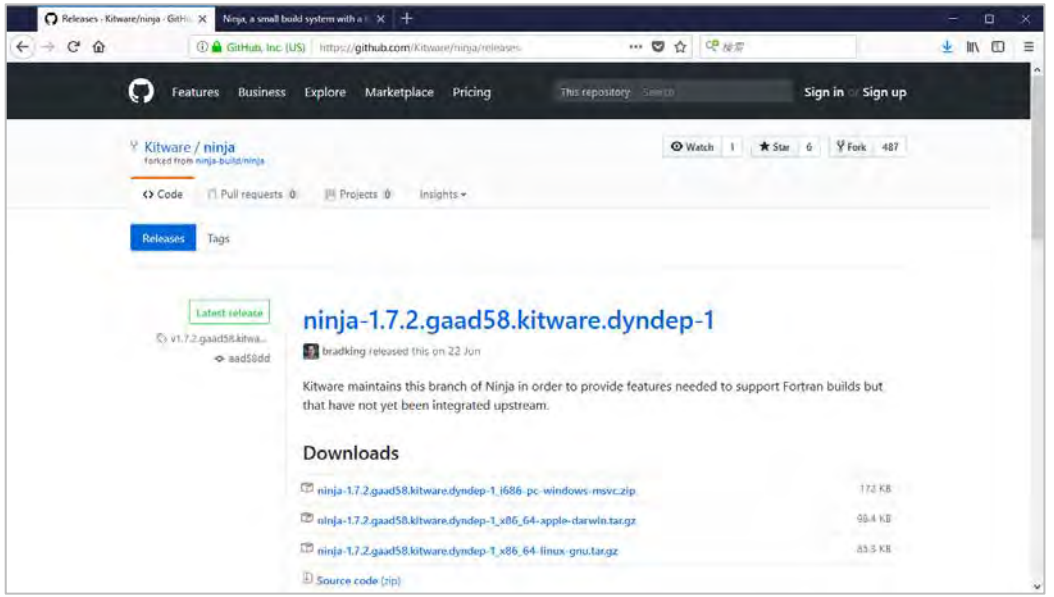
Intel Core i7-6700 3.40GHz * 4コア相当のマシンで計測
time コマンドで3回の平均値

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

帝京大学 TEIKYO

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。

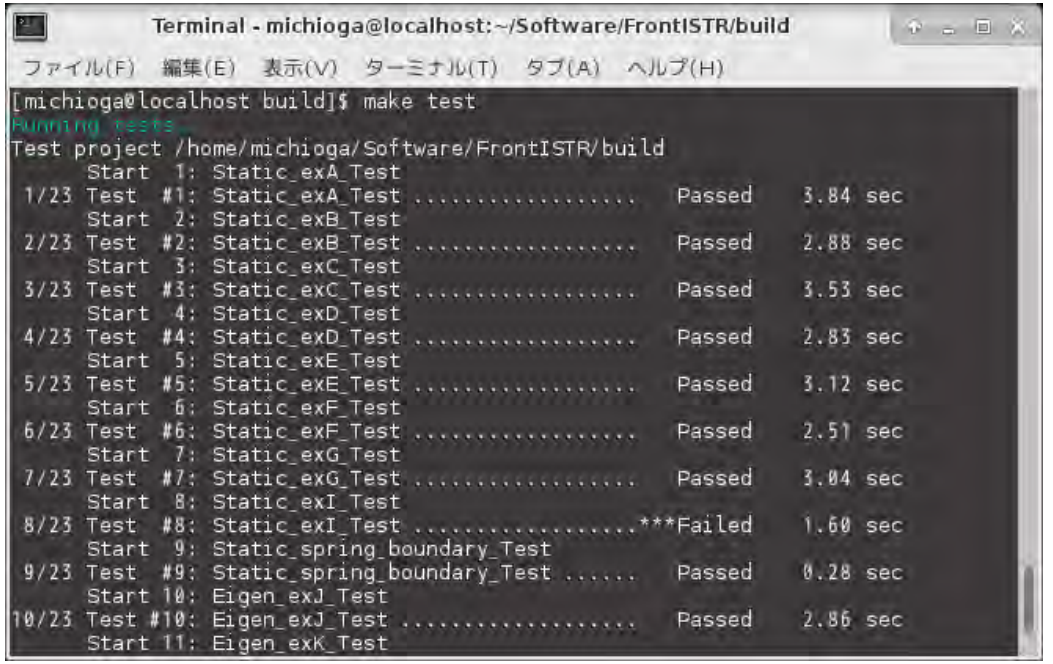


帝京大学 TEIKYO

簡易テスト機能

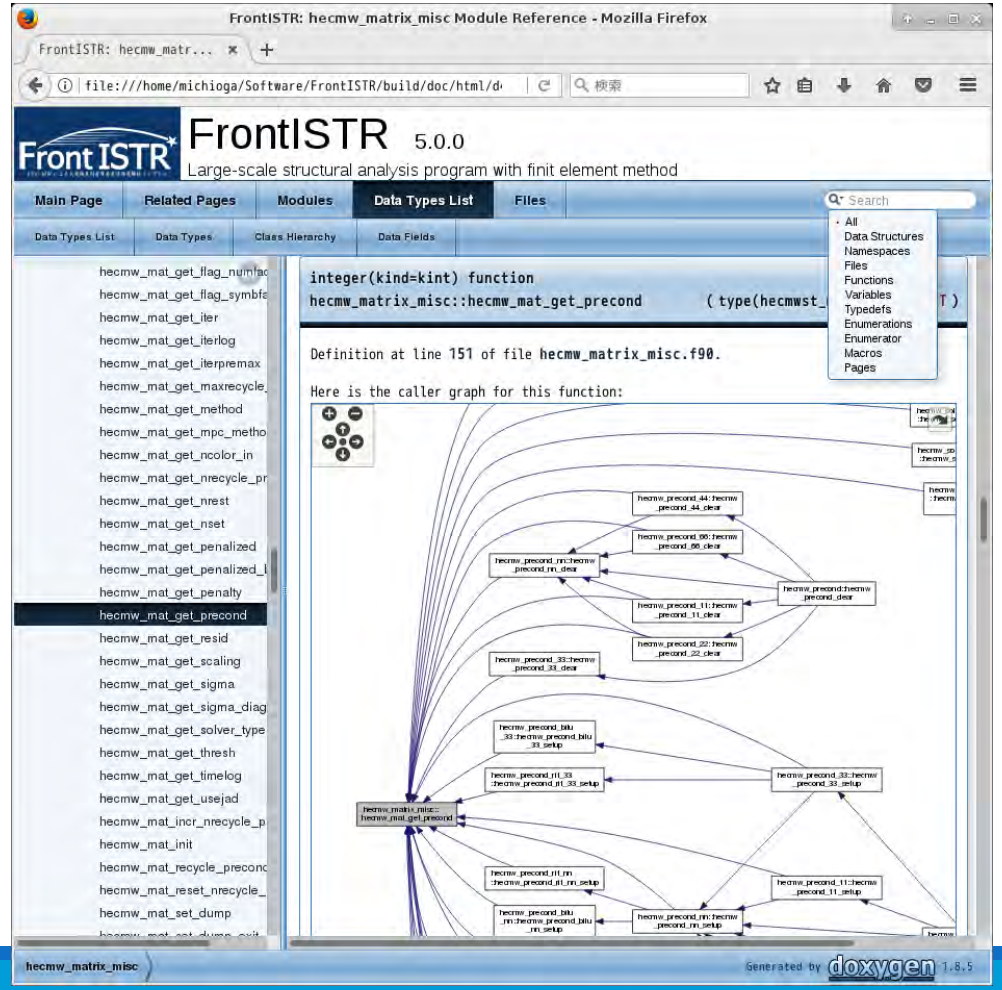
CMakeに簡易テスト機能が統合されました

- 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。



帝京大学 TEIKYO

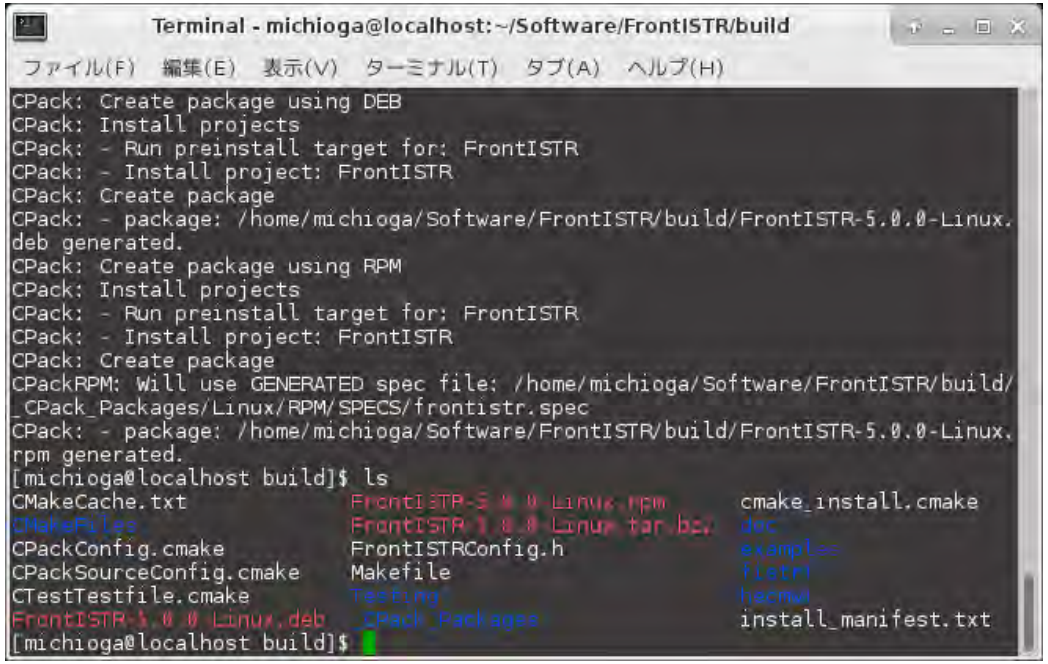
ソースコードのドキュメンテーション



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。

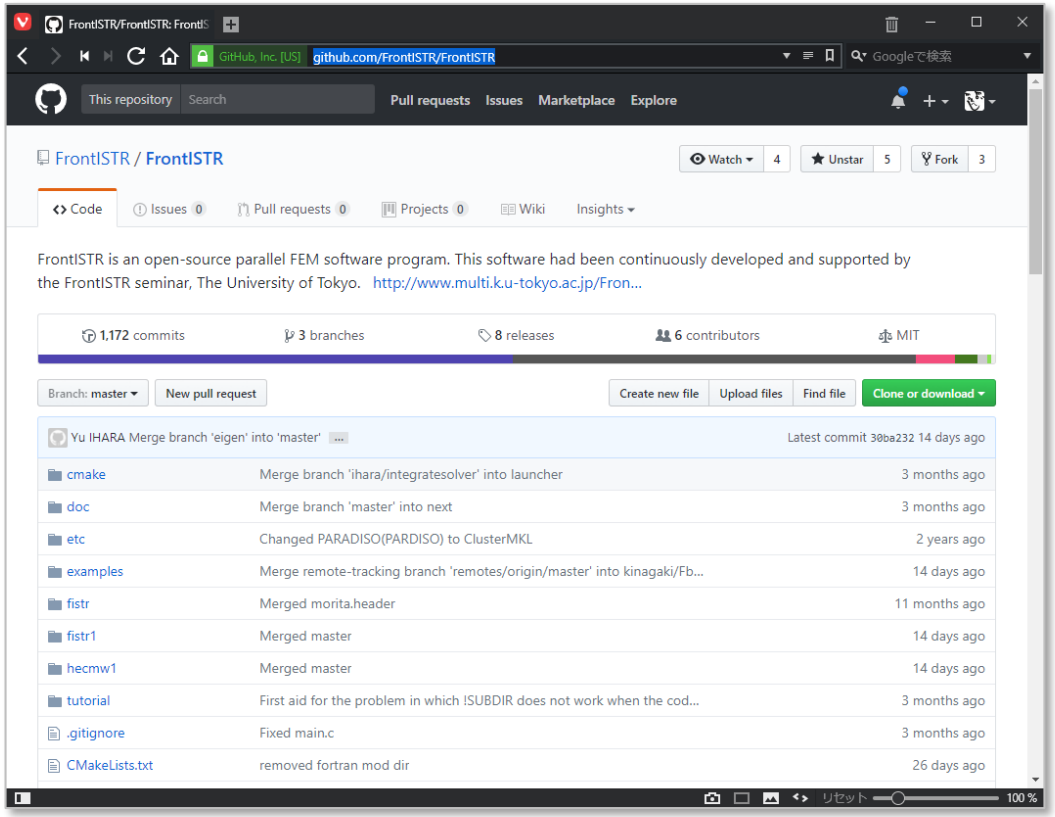
帝京大学
TEIKYO

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。



帝京大学
TEIKYO

ご清聴ありがとうございました