FrontFlow / violet Cartesian

Kenji ONO, AICS, RIKEN
keno@riken.jp

2012 年 10 月 4 日

1 ディレクトリ・ファイル構成 (Directories and files)

ffvc-x.x.x.tar.gz を解凍すると，以下のようなファイル構成になります[*1].

```
ffvc-x.x.x
  |
  +- BUILD                      アプリケーションのコンパイル方法のメモ
  +- COPYING                    コピーライト
  +- README                     最初に見るべきファイル
  +- RELEASE                    リリース情報
  |
  +- bin                        実行モジュールの配置ディレクトリ
  |
  +- doc                        ドキュメント
  |   +- ffvc_ug.pdf            FFV-C ソルバーのユーザガイド
  |
  +- doxygen                    Doxygen ドキュメントディレクトリ
  |   +- FFV                    FFV クラスのドキュメント
  |   +- Conf                   Doxygen ファイルを生成するための設定ファイル
  |   +- FB                     FB クラスのドキュメント
  |   +- IP                     Intrinsic クラスのドキュメント
  |
  +- example                    例題
  |   +- Cavity_binary          三次元のキャビティフロー例題（バイナリモデル）
  |   +- Cavity_cut             三次元のキャビティフロー例題（カット情報モデル）
  |   +- LDC                    辺長比 1:1:2 のキャビティフロー例題（実験値との比較）
  |   +- PMT                    性能測定用例題
  |   +- Sphere                 球周りの流れ例題
  |
  +- src                        ソースコードディレクトリ
      +- Cutlib-x.x.x           カットライブラリ
      +- F_CORE                 Fortran のコアプログラム（Binary 版）
      +- F_VOF                  VOF クラスの Fortran ファイル
      +- FB                     FlowBase クラス（ユーザー定義クラス群）
      +- FFV                    FFV-C ソルバ
      +- IP                     組み込み例題クラス群
      +- PMlib-x.x              性能測定ライブラリ
      +- Polylib-x.x.x          ポリゴン管理ライブラリ
```

[*1] doxygen ディレクトリについては，doxygen ファイルを生成するために必要な設定ファイルのみを提供しています．Conf ディレクトリ内で make を実行すると各ディレクトリに doxygen ファイルが生成されます．

2 ビルド手順 (How to build)

ビルドは，OpenMPI, TextParser, CPMlib, FFV の順序で行います．

2.1 必要な外部ライブラリ (External libraries)

次のライブラリが必要です．TextParser, CPMlib については，本パッケージ内に同梱しています．

- OpenMPI (1.6.2)
- TextParser (1.1)
- CPMlib (1.0.6)

2.2 コンパイラ，ビルドツール (Compiler and build tools)

- コンパイラ
  利用するコンパイラは，Intel Compiler C/C++, Fortran XE Intel(R) 64, Version 13.0.0.088 を利用しています．gcc, gfortran, xlc/c++, xlf でのコンパイルも可能です．
- ビルドツール
  make, auto tools を利用します．一部 autotools が使えない場合には，手動による make で対応します．

2.3 ビルド方法 (Build)

簡単な手順を以下に示します．詳細については，`ffvc_ug.pdf` のインストールをご覧ください．

1. TextParser のインストール
   `TextParser-x.x.tar.gz` を展開し，トップディレクトリの `config_tp.sh` を実行します．

```
$ configure_tp.sh /usr/local/TextParser
$ make
$ sudo make install または make install
$ make distclean
```

2. CPMlib のインストール

   `CPMlib-x.x.x.tar.gz` を展開し，トップディレクトリの `config_cpm.sh` を実行します．

```
$ configure_cpm.sh /usr/local/cpm
$ make
$ sudo make install または make install
$ make distclean
```

   configure がうまくいかない場合は，`Makefile_hand` を使い，コンパイルします．

```
$ make -f Makefile_hand mpi
```

3. FFV-C のコンパイル

   (a) 一括コンパイル

Polylib, Cutlib, PMlib が FFV ディレクトリと同じ階層に配置されていることが必要です.

```
$ make depend
$ make
```

(b) 個別にコンパイル

以下の順序でコンパイルを行います．コンパイルをやり直す場合には,`make clean`, `make allclean` を実行します.

i. Polylib

`Makefile` のマクロ変数を指定し，コンパイルします.

```
$ make depend
$ make
```

ii. Cutlib

`Makefile` のマクロ変数を指定し，コンパイルします.

```
$ make depend
$ make
```

iii. PMlib

`Makefile` のマクロ変数を指定し，コンパイルします.

```
$ make depend
$ make
```

iv. FFV-C

`make_setting` のマクロ変数を指定し，コンパイルします.

```
$ make depend
$ make
```

3 実行手順 (How to run)

3.1 入力ファイル、設定ファイル (Input files)

実行に必要なファイルは計算対象とする問題により異なりますが，組み込み例題の場合にはパラメータファイルのみです．パラメータファイルの例が，example 配下のディレクトリにあります．パラメータファイルの指定の詳細については，ユーザガイドをご覧ください.

3.2 出力ファイル (Output files)

FFV-C ソルバーを実行すると，**表 1** に示すファイルが生成されます．また，log ファイルについては，Log セクションで生成の有無を指定します．全ての出力ファイルは，パラメータファイルにより指定できます.

3.3 サンプルデータ (Sample data)

サンプルデータとして，example ディレクトリ配下にある PMT をご利用ください．PMT の例題は，並列性能の測定を行う例題でファイルは出力しません．反復法の上限値を変更することにより，実行時間を調整することが出来ます．サンプルの入力ファイルでは，Intel Xeon Core i7 2.66GHz (1thread) で 320 秒程度の実行時

表 1 実行時に生成されるファイル

カテゴリ	ファイル名	出力内容
解析条件情報	condition.txt	計算条件，前処理，ソルバー起動時のログ
領域情報	DomainInfo.txt	並列計算時の計算領域の分割に関する情報
性能情報	profiling.txt	実行時間サンプリング出力ファイル
基本履歴	history_base.txt	ステップ数，時刻，反復回数，収束状況などの情報
コンポーネント履歴	history_compo.txt	内部境界のモニタ情報
流量収支履歴	history_domainflux.txt	計算外部領域における流入出流量，平均速度の情報
反復履歴	history_iteration.txt	反復解法の収束履歴
サンプリング履歴	sampling.txt	サンプリング指定時の出力ファイル
壁面情報履歴	history_log_wall.txt	壁面に関する情報の履歴
瞬時値データ	vel_*.sph	速度の瞬時値
	prs_*.sph	圧力の瞬時値
	tmp_*.sph	温度の瞬時値
平均値データ	vela_*.sph	速度の時間平均値
	prsa_*.sph	圧力の時間平均値
	tmpa_*.sph	温度の時間平均値
派生データ	tp_*.sph	全圧
	vrt_*.sph	渦度
	hlt_*.sph	ヘリシティ
	i2vgt_*.sph	速度勾配テンソルの第 2 不変量

間です.

3.4 実行方法 (Run)

実行方法は，pm_1.tp を入力パラメータファイルとして，以下のようになります.

```
$ ffvc pm_1.tp
```

3.5 参考情報 (Reference information)

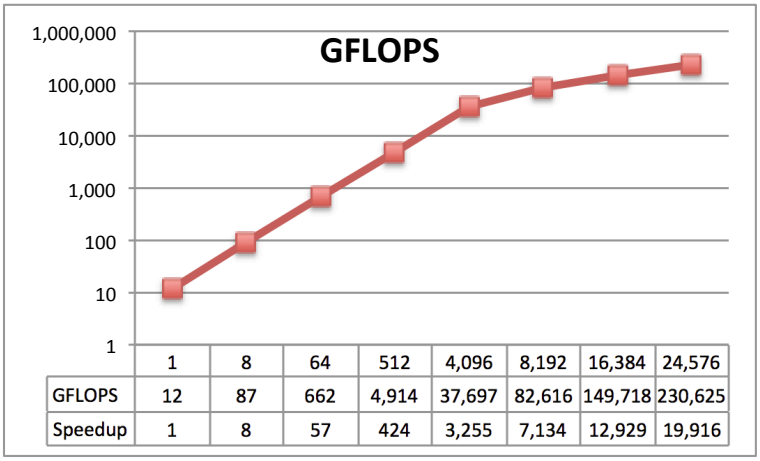
下記の Global_Voxel の要素数を変更することにより，メモリサイズ，ファイルサイズ，計算時間が変わります.

```
DomainInfo {
  Unit_of_Length  = "Non_Dimensional"
  Global_origin   = (-0.5, -0.5, -0.5   )
  Global_region   = (1.0,  1.0,  1.0    )
  Global_voxel    = (128   , 128   , 128   )
  ActiveSubDomain_File = ""
}
```

上記の分割数での weak scaling の結果を**図 1** に示します．実行時間は 140-160 秒の範囲です.

4 検証手順 (How to validate)

暫定的な検証として，example/PMT/history_base.txt の内容と diff をとって比較してください.



	1	8	64	512	4,096	8,192	16,384	24,576
GFLOPS	12	87	662	4,914	37,697	82,616	149,718	230,625
Speedup	1	8	57	424	3,255	7,134	12,929	19,916

図 1 PMT クラスの weak scaling 実行性能.

4.1 検証プログラム (Test program)

(検証プログラムの内容説明)

4.2 検証方法 (Run test program)

(検証プログラムの具体的な実行方法、実行結果の解釈についての説明)