Gtool Update

Youhei SASAKI

2012 年 08 月 28 日

Outline

状況整理

dc_utils

gtool5

Outline

状況整理

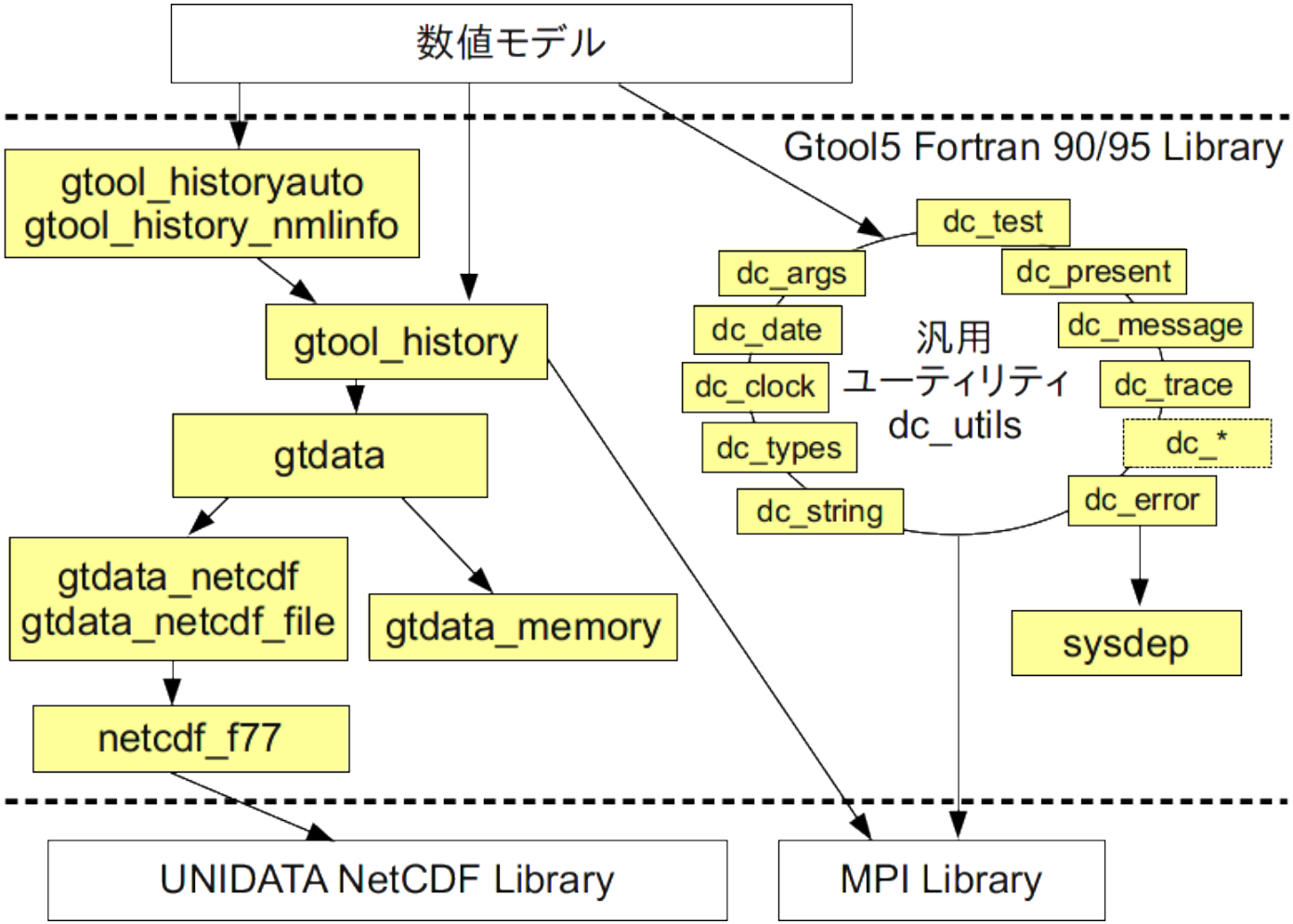
dc_utils

gtool5

開発

- ソースコード: CVS → Git へ
  - Read-Only: `http://www.gfd-dennou.org/library/gtool/git_repos/`
  - Full Access: `ssh://dennou-k.gfd-dennou.org/ftp/arch/gtool/git_repos/`
  - Web で履歴も見られるようにしたい
    - そういえば佐々木がなんかする筈だった気がする
- Web ページ
  - CVS コミットがトリガーになって色々走るようになっている
  - なので, これを変更したい (←そのうち)

gtool5 のデータ構造



TODO 一覧 (2012/05/15)

High Priority

- バグ潰し: `HistoryAuto` , `dc_clock` のメモリリーク
- 速度: 素の NetCDF API を使う場合に比して 1.5 〜 2 倍遅い
- 単なる `HistoryCreate` だと時間次元を `double` にできない

Low Proprity

- NetCDF-4 対応: MPI-I/O, Fortran90 interface を直接叩く
- `dc_` と `gt_` の依存の分離
- gtool4/NetCDF 規約と CF Convention との対応
- configure 周り
- Web の更新, 環境整備

やったこと + TODO 更新 (2012/08/28)

Done

- dc_ と gt_ の依存の分離
  - HistoryAuto , dc_clock のバグ潰しが目的
- configure 周り → done
  - dc_utils を分離した結果として, そうなった
- NetCDF-4 対応: Fortran90 interface を直接叩くようにした

TODO

- バグ潰し: HistoryAuto , dc_clock のメモリリーク
- 単なる HistoryPut だと時間次元を double で出せない (?) 問題への対処
- NetCDF-4 対応: MPI-I/O
- gtool4/NetCDF 規約と CF Convention との対応
- Web の更新, 環境整備

Outline

状況整理

dc_utils

gtool5

dc_utils: 汎用ユーティリティ

- Fortran90/95 用に文字列処理, エラー処理などの関数, 手続を提供
  - 大雑把に以下の関数を提供 (現在のチュートリアルより):
    - dc_sysdep (旧: sysdep): 機種依存の処理を吸収
    - dc_types: 種別型パラメタの提供
    - dc_args: コマンドライン引数解析
    - dc_message: メッセージ出力
    - dc_calendar: 暦および日時の操作
    - dc_clock: CPU 時間の計測
    - dc_trace: デバッグ出力
    - dc_error: エラー処理
  - 他に dc_string, dc_hash, dc_regex, dc_url など

dc_utils: 変更点

- 名前の変更
  - `sysdep` → `dc_sysdep`
- エラー番号の変更, 削除
  - `NF_...` は廃止 → NetCDF のエラー番号を使う
  - `GT_...`, `HST_...` は gtool 側へ移動
  - `DC_...` は -900 番台へ
- `NF_INT` 等の型の削除
  - gtool で `use netcdf` する
- 後方互換性
  - 現在確認してない (そのうち確認する).
  - 現在公開されている gtool5 にあるテストは全て通る

dc_utils: 問題点 (?)

- ソースコードの生成部分: **Ruby** が必要
  - `GETARG` or `GET_COMMAND_ARGUMENT`
  - `IARGC` or `COMMAND_ARGUMENT_COUNT`
  - `EIXT` or `STOP` (or `EXIT(3)`, `ERRTRA-SETRCD`, `SETRCD`)
  - `MAXDIM`
    - 処理系による配列最大次元 (7) の拡張を許すため

- ERROR 処理の際に system の `strerror` を呼ぶ必要がある
  - これまでは NetCDF の `nf_strerror` を使っていた
  - 直接 `strerror(3)` を呼ぶことにした
  - しかし `strerror(3)` が処理系依存: thread safe じゃないこともある.

- テスト, 可搬性の確保 (?)
  - ドキュメントの更新
  - 関数毎に Unit test 用意しなければならない
  - SX, SR, Windows(+ Intel Fortran?) でのテスト

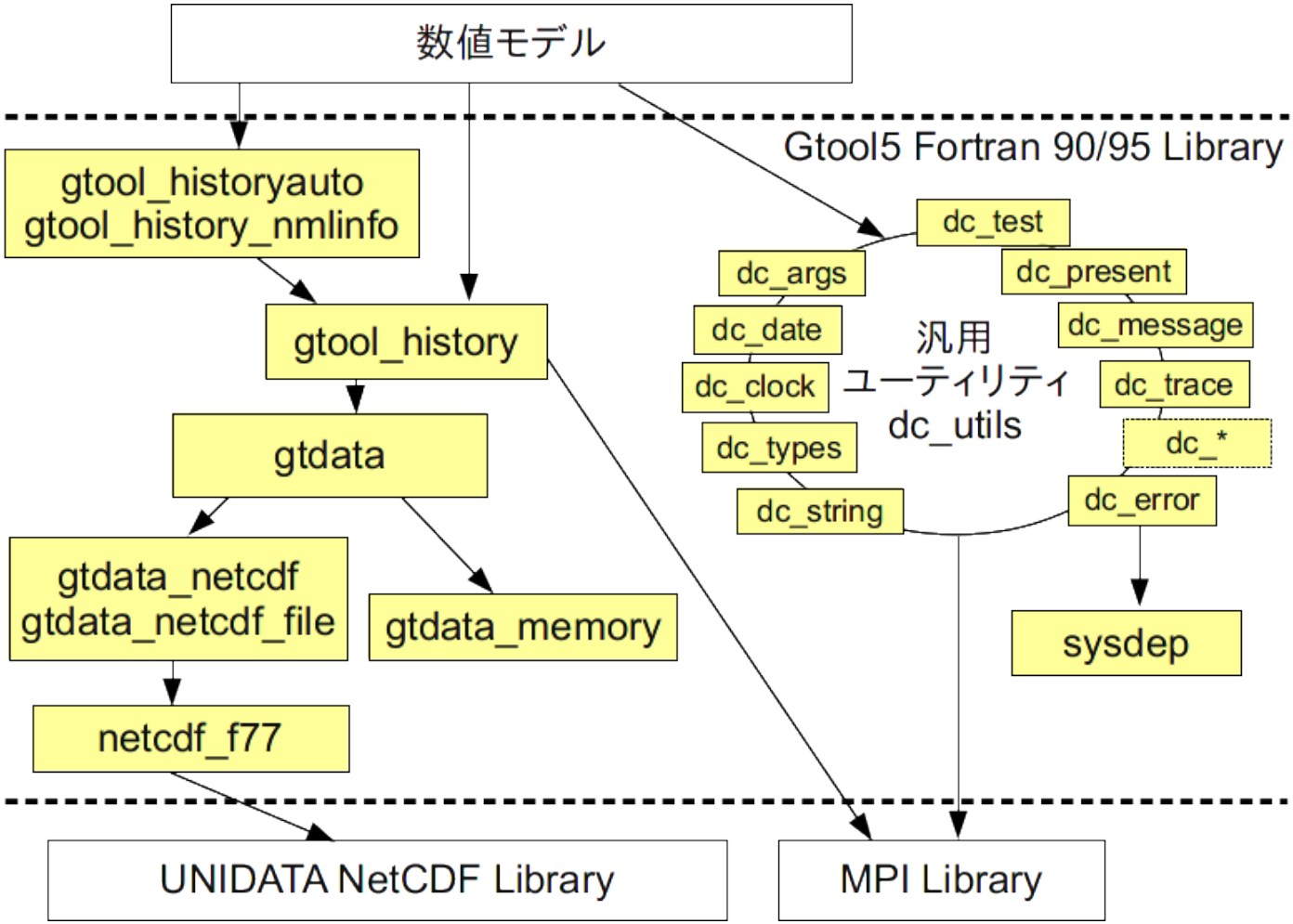
Outline

状況整理

dc_utils

gtool5

gtool5 のデータ構造 (再掲)



gtool5: 変更点

- gt_error, gt_url の追加
  - gt_error: GT_... と HST_... のエラーを扱うための dc_error の wapper
    - GT_... を -700 番台, HST_... を -800 番台へ移動
  - gt_url: 「ファイル名変数名」をパースするための関数
    - ... これ, 使ってますかね?

- netcdf_f77 の削除
  - 直接 NetCDF の F90 インターフェースを叩くようにした

- gtdata_memory の削除
  - 使ってない, 筈

- gt_data 層の gt_data_netcdf とのマージ
  - gt_data は NetCDF 以外の形式での I/O を提供するためのレイヤー
  - 使わない部分はバッサリ削除して, なるべく直接 NetCDF へ渡すように.

- 速度
  - 手元では Read で 40% 程度, Write で 30% 程度の高速化
  - 生 NetCDF と比較はこれから (手元だと違いが良くわからない).

gtool5: 問題点 (?)

- ソースコードの生成に Ruby が (略
  - 主に `MAXDIM` 関連
- ドキュメント, チュートリアルの更新
  - 既にチュートリアルは現状に即していない
  - 早々に書き変えたいのだけれど rdoc-f95 が動かない
- バグ (?)
  - `HistoryAuto` のメモリリーク: 構造体を整理中
- HSPACK をどうするか?
  - `HSPACK`: gtool_history の FORTRAN77 インターフェース
  - 潰せると保守は非常に楽なんですが. . .