外部計算資源の利用について

リリース ***1.0.1***

SIP-MI

2022 年 **04** 月 **15** 日

目次:

第 1 章

WFAS7 利用_FrontISTR 実行_検証版_外部計算_DPF ワークフロー

1.1 概要

本ワークフローは WFAS[*1] で行うクリープ性能試験で利用する FrontISTR 解析ワークフローを外部計算機資源のうち DPF 計算環境の利用による計算を使って実行するワークフローである。本ワークフローは WFAS7 利用_FrontISTR 実行_検証版を元に作られているため、以下の説明は省略する。

- ワークフローの説明
- 入力データのフォーマット
- 各ツールの説明
- 計算結果の確認
- 計算結果のファイル

詳細は WFAS7 利用_FrontISTR 実行_検証版の使い方を参照すること。ここでは外部計算機利用のための説明を行う。

[*1] WFAS は SIP-MI ラボで開発された溶接シミュレーションソフトウェアによる解析を WEB GUI から行えるようにしたアプリケーションである。

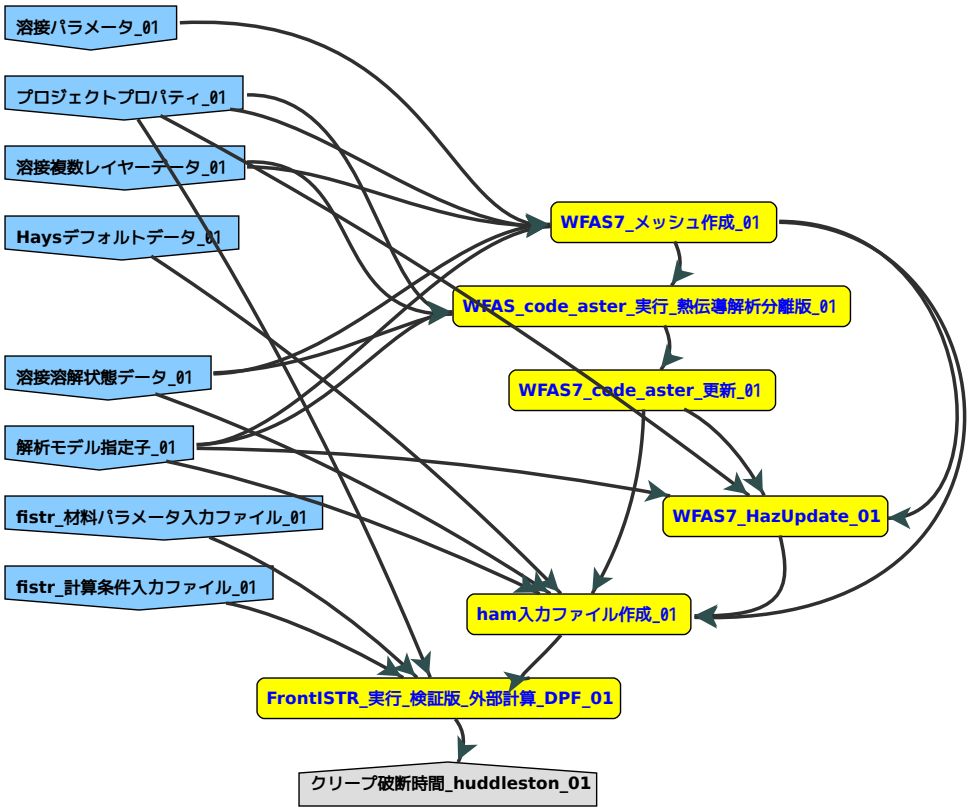

図 1.1　WFAS7 利用_FrontISTR 実行_検証版_外部計算_DPF ワークフロー

注釈: 本ワークフローは WFAS から利用することを前提条件としたワークフローを元にしているため、入力パラメータが特殊である。

注釈: 外部計算機として DPF 計算環境を利用するため、その手続きと準備が必要となる。

1.2 外部計算機利用の仕組み

本ワークフローで使う外部計算機利用は MInt システムと WebAPI を利用したポーリングシステムを利用して各モジュールの計算を MInt システムおよび NIMS の所外の計算機を利用して行うものである。以下この方式を WebAPI 方式とする。WebAPI 方式での外部計算機利用の実行イメージを (図 1.2) に示す。

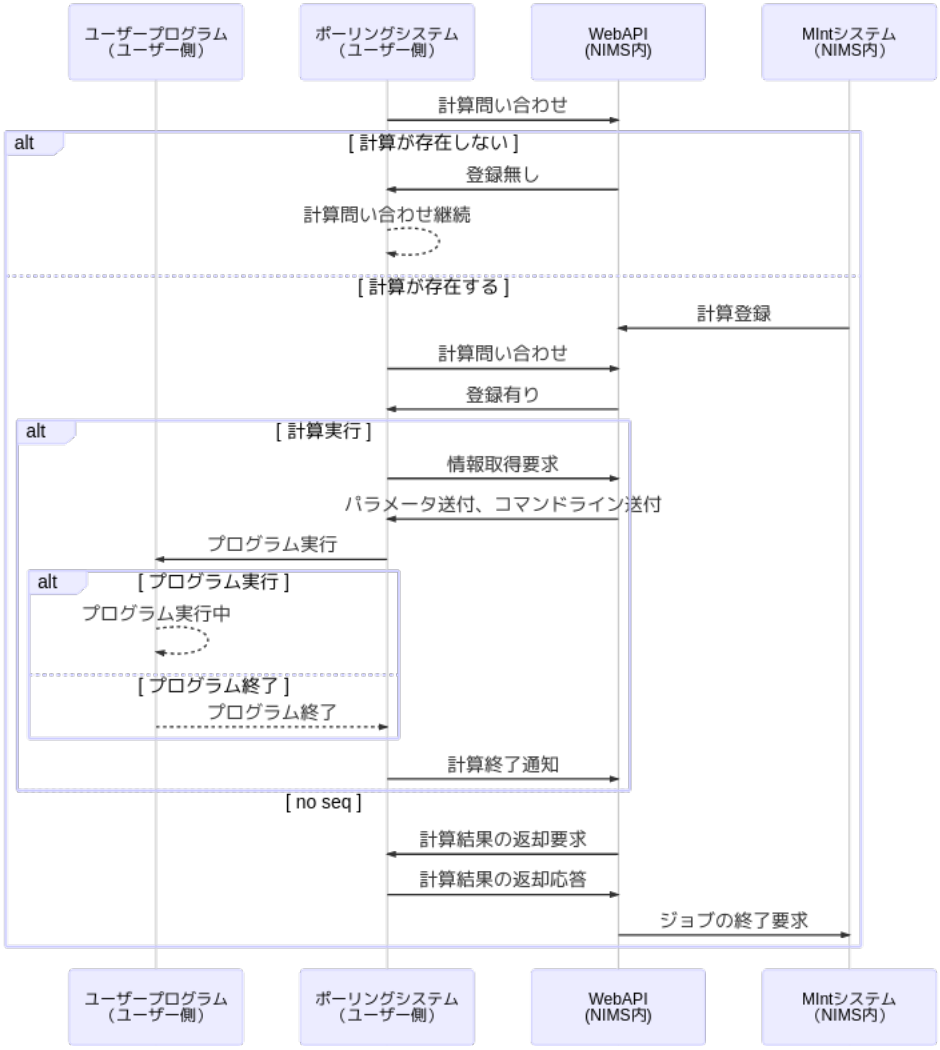

図 1.2　ポーリングシステムの流れ

1.3 外部計算機利用予測モジュールの処理

外部計算機利用予測モジュール（本ワークフローでは全ての予測モジュールが対象）内で外部計算機側の処理が実行されるまでの流れを下記に示す。

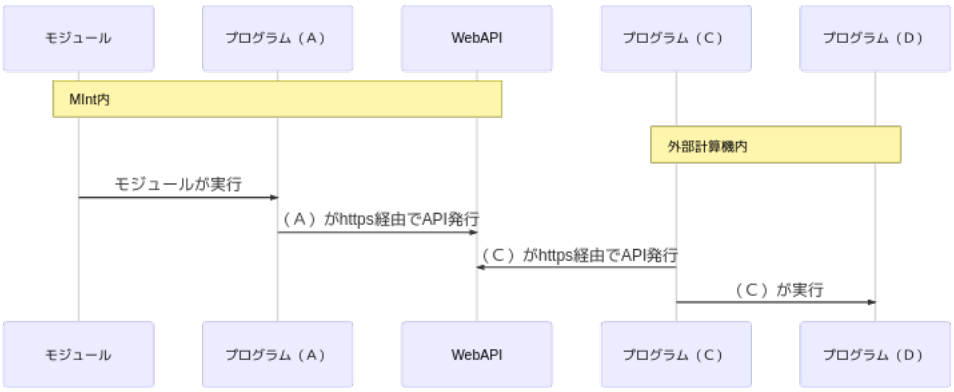

図 1.3　WebAPI 方式でのコマンドの流れ

- モジュール
  - MInt のワークフローシステムによって実行されるモジュール
  - ワークフロー作成者が用意する。
  - プログラム（A）を実行する
- プログラム（A）
  - モジュールによって実行されるプログラム。モジュールごとに任意の名前で用意する。
  - ワークフロー作成者が用意する。
  - モジュール固有の前処理を行う。
  - モジュール内で**misrc_distributed_computing_assist_api/debug/mi-system-side/mi-system-wf.py** を実行する。
    - WebAPI へ計算の情報が登録される。
    - 以降このプログラムが外部計算機資源側（以下の（C））と API を介して計算に必要な処理を行う。
- WebAPI (このプログラムが MInt システムと外部計算機との通信を中継する。)
  - 外部計算の準備を行う。
    - 送受信するファイルはパラメータとしてあらかじめ設定しておく。
  - WebAPI 経由で（C）からのアクセスを受け付ける
  - （A）から計算の情報登録が無い限り、（C）からアクセスがあっても計算は始まらない。
  - ワークフローを実行したユーザーのトークンと（C）からのトークンが合致しないと（C）は適正な通信相手とならない。
- プログラム（C）
  - ポーリングプログラムである。
  - 外部計算機資源（ユーザー）側で用意する。
  - **misrc_distributed_computing_assist_api/debug/remote-side/mi-system-remote.py** を実行する。
  - 外部計算機上で実行するプログラム名は、このプログラム経由で MInt システムから受信され、このプログラムが実行する。

  - 認証情報はこのプログラム（C）が使用する。認証情報が無いと WebAPI にアクセスできない。詳細は 1.4 章の利用の準備 で説明する。
- プログラム（D）
  -（C）から実行される外部計算用スクリプト。
  - 外部計算機資源（ユーザー）側で用意する。
  - 名前は任意。（プログラム（C）経由で伝えられるため、あらかじめ MInt システム側に設定が必要）

1.3.1 処理の流れ

ラン実行によるモジュールの処理の流れは以下のとおりとなる。

- クライアント（DPF）でポーリング開始
- MInt システム（予測モジュール）から API サーバーへ外部計算用計算の登録
- クライアントで API サーバーから外部計算用計算情報の受信と詳細取得
- クライアントで API サーバーからパラメータの受信
- クライアントで FronISTR の処理
- クライアントで API サーバーへ FrontISTR 処理の終了通知と結果の送信
- MInt システム（予測モジュール）で API サーバーから結果の受信

1.4 利用の準備

本ワークフローは FrontISTR 計算を DPF が用意する計算機群を使用して計算を行うが、MInt チームが用意したアカウントを利用する場合、ワークフローも既存のものが使用可能である。DPF に専用のアカウントが必要な場合は以下の準備が必要となり、ワークフローも専用のものを用意する必要がある。

- DPF 計算環境を利用するためのユーザーおよび資源
  - DPF の資料にしたがって準備する。
- DPF 計算環境でユーザー個別の FrontISTR のインストール
  - インストールは DPF の資料にしたがうが、FrontISTR のインストール資材は MInt システムチームから入手する。
- DPF 計算環境で外部計算機資源を WebAPI で利用するための資材の展開
  - こちらは通常の外部計算機資源の利用マニュアルに従って入手、展開する。
- 利用申請
  - API トークンの外部計算機利用ワークフロー利用申請
- 認証情報
  - API 認証情報の取得
- 環境構築
  - 利用者サイドの計算機および計算機資源。
- ポーリングの開始

1.4.1 DPF を利用する場合の注意事項

DPF にユーザー専用のアカウントを作って利用する場合、以下の点に注意すること。

- WebAPI のポート番号：50443 で通信ができないので、port パラメータで 443 を利用する。
- 認証局ファイル：WebAPI 通信の際、proxy 経由に必要な認証局ファイルの指定が必要である。no_auth パラメータで指定する。
- 作業ディレクトリの指定：通常、/tmp を使うが、DPF では/home/<ユーザーディレクトリ>/<作業ディレクトリ名>を用意し、calc_base パラメータで指定する。

注釈: 認証局ファイルは ICT 室から入手できる nims_proxy.cer を利用する。

1.4.2 利用申請

WebAPI 方式を利用したポーリング実行に必要な利用申請を行う。このタイプのワークフローはユーザー専用となるため、その手続きが必要である。申請後、専用ワークフローのワークフロー ID が返送されるので、実行にはそのワークフロー ID を利用する。

1.4.3 認証情報の準備

MInt 側担当者と打ち合わせて下記の情報を用意する。

- ホスト情報
  - MInt 側で API の発行者を識別するための文字列。利用者が持つドメインとホスト名を使うのが一般的であるが、

同じである必要は無い。* API トークン

- MInt の API 認証システムを使用するためのトークン。すでに MInt システム利用申請を済ませ、ログイン ID とパス

ワード入手済であれば、MInt システムログイン後、ユーザープロファイル管理システムメニューで表示される、「API トークン」を

  - ユーザーの環境でポーリングスクリプトを動作させるときに必要であるが、後述のログイン方式を利用する場>合はトークン自体は必要ない。
- MInt の URL
  - MInt の URL(エンドポイントは不要) を、MInt 運用チームに問い合わせておく。
- WebAPI 方式を利用できるように MInt 運用チームに設定を依頼する。
  - ホスト情報と使用コマンド情報を MInt システムの WebAPI に事前登録する。

1.4.4 環境設定

利用申請を行うと、実行に必要な環境構築方法や資材の入手方法のマニュアルが返送されるので、それを使用して環境構築および資材を入手、展開、実行準備を行っておく。

1.4.5 ポーリングの開始

認証情報と共にポーリングプログラムを動作させておく。事前に設定した情報に従って MInt システム側と通信し、入力ファイルの受信、計算、出力ファイルの送信が自動的に行われる。認証情報が無い、間違っている、などの場合は>ポーリングは失敗し、計算は行われない。また **ワークフローを実行したユーザーと同じユーザーのトークンまたはログイン方式での同じユーザー** で実行しないとこちらもポーリングは失敗し、計算は行われない。

1. 以下 2. または 3. のどちからの方法で、**mi-system-remote.py** を実行する。
2. トークン指定方式

```
$ cd <展開した場所>/misrc_distributed_computing_assist_remote_side
$ python mi-system-remote.py <ホスト情報> https://nims.mintsys.jp <API token>␣
↪port:443 no_auth:/home/misystem/nims_proxy.cer calc_base:/home/misystem/remote_
↪calculation
site id = <ホスト情報> で指定した識別子
base url = https://nims.mintsys.jp:443
```

3. ログイン方式

```
$ cd <展開した場所>/misrc_distributed_computing_assist_remote_side
$ python mi-system-remote.py <ホスト情報> https://nims.mintsys.jp login port:443 no_
↪auth:/home/misystem/nims_proxy.cer calc_base:/home/misystem/remote_calculation
site id = <ホスト情報> で指定した識別子
base url = https://nims.mintsys.jp:443
nims.mintsys.jp へのログイン
ログイン ID: <MInt システムのログイン名>
パスワード: <同、パスワード>
 token = <ログイン名のトークンの表示>
...
```

注釈: no_auth パラメータで NIMS の用意した認証局ファイルを指定する必要がある。また port パラメータで 443 を指定する必要がある。

注釈: ホスト情報と API token は `get_authorizaion_infomation` の get_authorizaion_infomation で入手したそれぞれの認証情報を指定する。

注釈: 通常は 50443 ポートを利用して行われるが、ポリシーによって使用不可能な場合はコマンドラインに port:443 とすることで 443 ポートでの通信も可能となる。

1.5 ワークフローの実行

DPF に専用のアカウント作った場合は専用のワークフローとなるので、利用可能になると専用の ID が送付される。実行はその ID で実行する。そうでない場合は ID、W0001100000000724 のワークフローを使用する。

1.5.1 ワークフロー実行中

実行中のワークフローを 図 1.4 に示す。

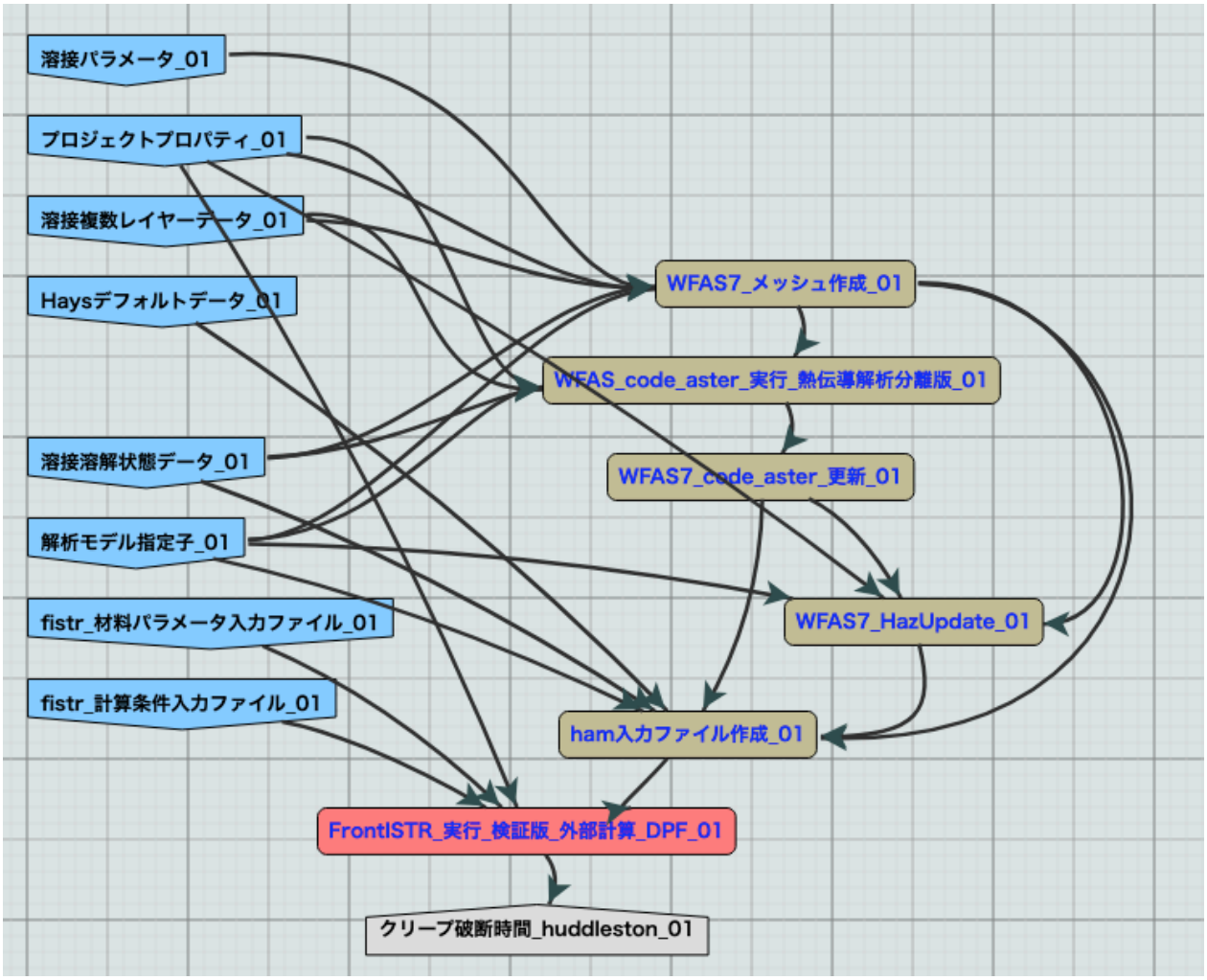

図 1.4　実行中のワークフローのイメージ

- 灰色になっているのが実行終了したモジュール
- 赤色になっているのが実行中のモジュール
- 黄色いのは未実行のモジュール (この例では全モジュールが実行済みか実行中なので無い)