外部計算資源の利用について

リリース ***1.0.1***

SIP-MI

2022 年 **03** 月 **18** 日

目次:

第 1 章

はじめに

MInt には、ワークフローを構成するモジュール内の一部分の処理を、MInt の計算ノード以外の「外部計算機」に行わせる機能がある。本機能によって、ユーザには下記に挙げる利点がある。

- 秘匿プログラムの使用
- 秘匿データの使用
- 特殊構成 (MInt の計算ノードでは対応できない) の計算機を使用できる
- 商用ソフトの使用 (MInt の計算ノードにも商用ソフトがインストールされているが、ライセンスの規定上、ほとんどの場合 NIMS 外からは利用できない)

外部計算資源の利用に際しては、MInt、外部計算機の双方が後述のセキュリティ水準を満たす必要がある。

1. MI コンソーシアム会則（秘密保持誓約書含む)、MInt システム利用規定など、MInt 利用に関わる契約・規定を遵守すること。
2. MInt 側は、下記のセキュリティ対策を実施すること。
   - 第三者による MInt のセキュリティ分析・セキュリティリスク診断を実施し、リスクを避ける設計を維持すること。
   - MInt を構成するサーバの OS・ミドルウェア・ライブラリ等に対し、継続的に脆弱性データベースを確認し、必要なアップデートを実施すること。
   - 不正アクセス監視やネットワーク負荷監視を実施すること。
3. 外部計算機側は、外部計算機として利用されるコンピュータに対し、十分なセキュリティ対策を実施すること。継続的に利用する場合には、定期的に対策状況を確認し、セキュリティレベルを維持すること。

外部計算資源利用には、SSH 方式と WebAPI 方式がある。前者は MInt から外部計算機へ SSH で必要なデータとコマンドをプッシュする方式である。単純で、外部計算を遅延なく開始できる利点があるが、外部計算機側で MInt に対し SSH のポートを開放してプッシュを受け入れる必要がある点は、特に企業ユーザではハードルが高いことが想定される。これに対し、後者は数分間隔で外部計算機側から MInt に WebAPI(https) でポーリングし、処理すべき案件が存在した場合は、必要なデータとコマンドがプルされる方式である。この方式では外部計算機側にポート開放の必要が無いが、外部計算の開始までにポーリング間隔に相当する遅延が生じる。

本機能は外部計算機内にユーザが持つ秘匿データの扱いにも十分な配慮が行われている。まず、ユーザが持つ秘匿データに関して、MInt が収集する情報はワークフローの各モジュールの入出力ポートの情報のみであり、モジュールの内部で完結する本機能のために、モジュールと外部計算機の間で送受信される情報は収集対象外である。外部計算機側は、MInt にあらかじめ定められたコマンドのみ実行できるように設定することができる。また、MInt に処理結果を返送する前に不必要なデータをワーキングディレクトリから削除することができる。

上記の機構によって、安全な外部計算が保証される。下記の各章で具体的な利用方法について記す。また、外部計

算資源の利用に際して本書では不明な点は、ユーザと MInt 運用チームとの協議で決定するものとする。

1.1 本書の対象

1.1.1 対象読者

本文書は既に MInt システムのユーザーであり、これから外部計算資源を利用したワークフローの実行を行おうとしている方 (以下利用者とする）を想定して記述しています。
対象読者は以下の知識を有しているか、知識を有している担当者が存在していることを前提としています。

- Linux(主に CentOS 7.x) の仕組みやコマンドについての基本的な知識
- オープンソースソフトウェアを扱う、基本的な知識
- MInt システムのワークフローの操作、ランの実行などの基本的な知識

1.1.2 対象範囲

本文書ではこれから外部計算機資源を利用した既存のワークフローの実行を行うための情報を元に記述されている。MInt システムに存在しない外部計算機資源を利用することを想定した予測モジュールやワークフローの新規作成に関する情報は記述しない。また外部計算機資源として利用者が用意する計算機 (以下外部計算機と言う) で動作するプログラムは NIMS 側での動作実績のあるものを対象とし、新規作成した上で外部計算機資源として利用する場合の情報も記述しない。（注：これはら別途「」に記述する。）
この他、外部計算機に展開する予測モジュール用の資材の詳細についてはその予測モジュールまたはワークフローの利用者マニュアルを参照することとし、本書では扱わない。

第 2 章

概要

2.1 利用イメージ

外部計算資源の利用イメージを (図 2.1) に示す。

- MInt は NIMS 構内の DMZ[*1] に存在する。
- ユーザは MInt 上に、外部計算を利用するモジュールを含んだワークフローを持つ。当該モジュールやワークフローの参照・実行権限は自機関内などに限定できる。
- ユーザが当該ワークフローを実行すると、外部計算を利用するモジュールで一部の処理が外部計算機に受け渡される。
- 外部計算機は処理の過程で、MInt に置けないデータやプログラムにアクセスできる。これらのアクセスを外部計算機の内部で完結させることで、安全な利用が可能となる。

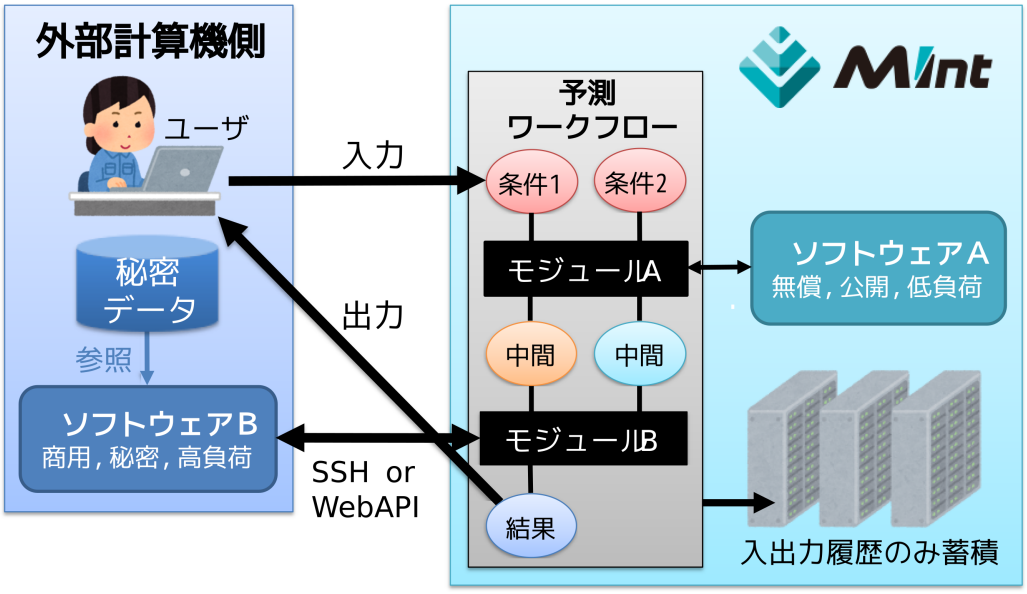

図 2.1　外部計算資源の利用イメージ

[*1] 物理的には NIMS 構内のサーバ室に存在するが、ネットワーク的には機構内 LAN とインターネットの双方からファイアウォールで切り離された領域。

2.2 SSH 方式と WebAPI 方式の比較

- SSH 方式
  - MInt から SSH で外部計算機にアクセスし、必要なファイルとコマンドをプッシュし、コマンドを発行し、結果を得る。
  - ファイルは内部で rsync -av を利用して送受信され、サイズは無制限である。
  - コマンドラインなどの文字列は Base64 エンコード無しで送受信される。
  - 外部計算機側 SSH サーバのポート (TCP/22 以外でも可) のインバウンドアクセスの開放が必要である。
  - 通信障害には弱い。計算中などで通信が切れると復帰できず、本バージョンでは実行プロセスが簡易なため計算続行が不可能である。
- WebAPI 方式
  - 外部計算機から MInt の WebAPI サーバにポーリングを行い、要処理案件の有無を確認する。ポーリング間隔は数分程度を想定している。案件があれば必要なデータとコマンドをプルし、自らコマンドを実行し、API で結果を送信する。
  - ファイルは Base64 エンコードされ、サイズはエンコード後に 2GiB[*2] 未満である必要がある。
  - コマンドラインなどの文字列は Base64 エンコード無しで送受信される。
  - MInt の WebAPI サーバへの https(TCP/50443) のアウトバウンドアクセスの許可が必要である。
  - サーバー側クライアント側で状態を保持しているため、通信障害が起きても再接続と計算続行が可能である。

[*2] GiB はギビバイトといい、コンピュータの容量や記憶装置の大きさを表す情報単位の一つである。1GiB は 2 の 30 乗バイトであり、1,073,741,824B である。

第 3 章

動作原理

3.1 SSH 方式

3.1.1 動作イメージ

SSH 方式での外部資源利用のイメージを (図 3.1) に示す。

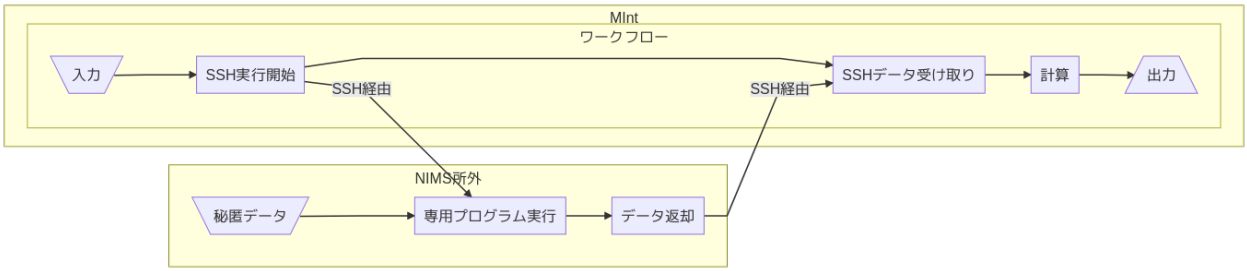

図 3.1　SSH 方式の外部資源利用のイメージ

3.1.2 ワークフロー例

SSH 方式の外部資源利用を含むワークフローを、MInt のワークフローデザイナで表示した例を示す。赤枠の部分が遠隔実行の行われるモジュールである。なお、本ワークフローは動作検証用サンプルとして、4 章の利用のための準備 で説明するインストール資材に含まれている。

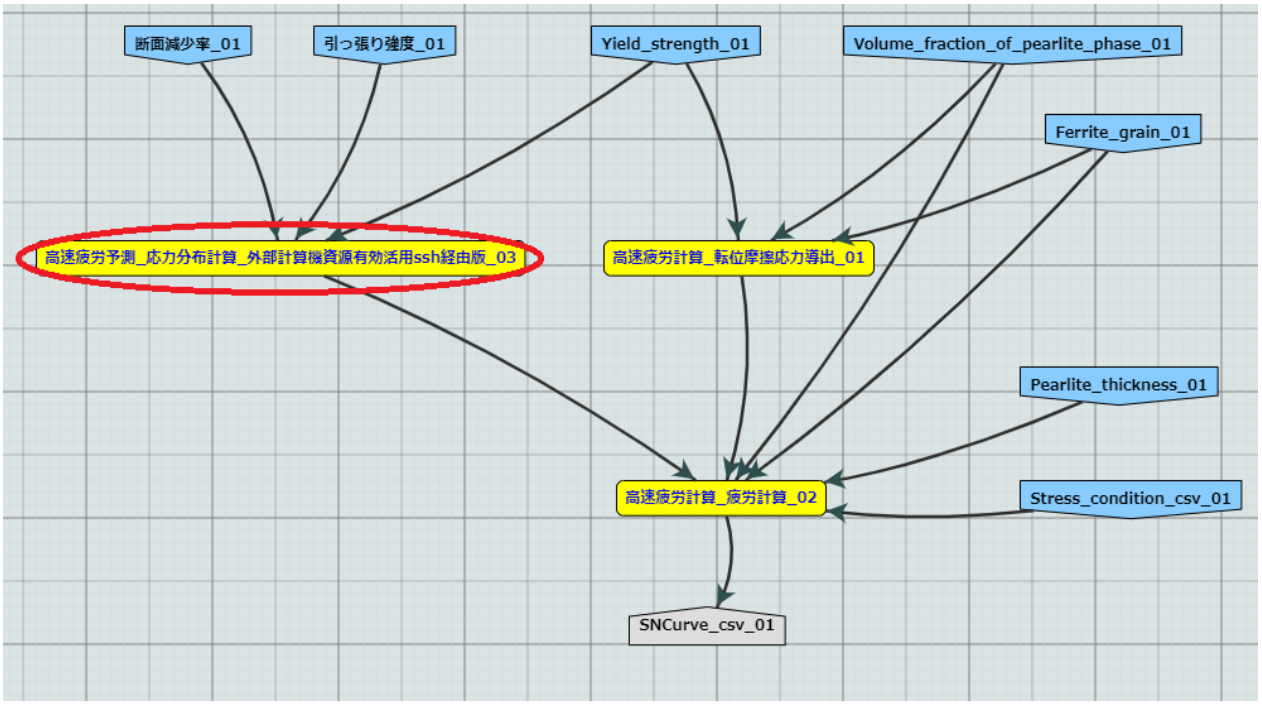

図 3.2　動作検証用のワークフロー

3.1.3 モジュール内の処理

外部資源利用を行うモジュール内で、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。

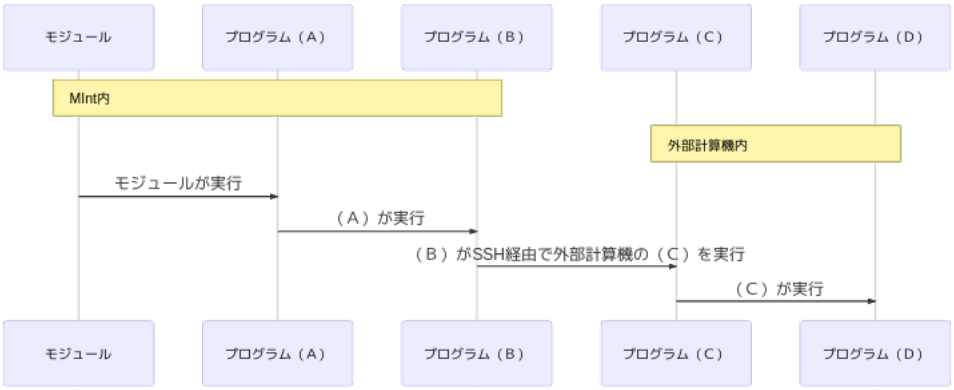

図 3.3　SSH 接続経由によるコマンド実行の流れ

- 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
  - 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
  - モジュール固有の前処理を行う。
  - モジュールごとに任意の名前で用意する。
  - 4 章の利用のための準備 で説明する編集を行う。
  - （B）を実行する。
- プログラム（B） このプログラムが外部計算機と通信を行う。
  - 外部計算の準備を行う。
  - 名前は任意の名前を使用可能。
  - テンプレートは **execute_remote_command.sample.sh** をコピーして使用する。
    - 4 章の利用のための準備 で説明する編集を行う。
    - （A）が実行するプログラム名とコピーしたプログラム名は同名としておく。
  - SSH 経由で（C）を実行する。
    - 送信するファイルはパラメータとして記述する。
    - 外部計算機上の一時ディレクトリ[*3] の内容を全部受信するため、MInt に送信しないデータは外部計算機側で（C）の実行終了前に削除する。
- プログラム（C）
  - 名前は プログラム（B）用のテンプレートで**execute_remote-side_program_ssh.sh** となっているが変更可能である。
    - 同名のテンプレートが用意されているので、複雑な処理を必要とする場合は、コピーして使用する。
    - （B）で実行されるプログラム名とコピーしたプログラム名は同名としておく。

[*3] 外部計算機では、処理は/tmp などに作成した一時ディレクトリで実行される。

- プログラム（D）
  - 外部計算機上のプログラムを（C）のみで完結させ、本スクリプト群は用意しない運用も可能である。

3.2 WebAPI 方式

3.2.1 動作イメージ

WebAPI 方式での外部計算の実行イメージを (図 3.4) に示す。

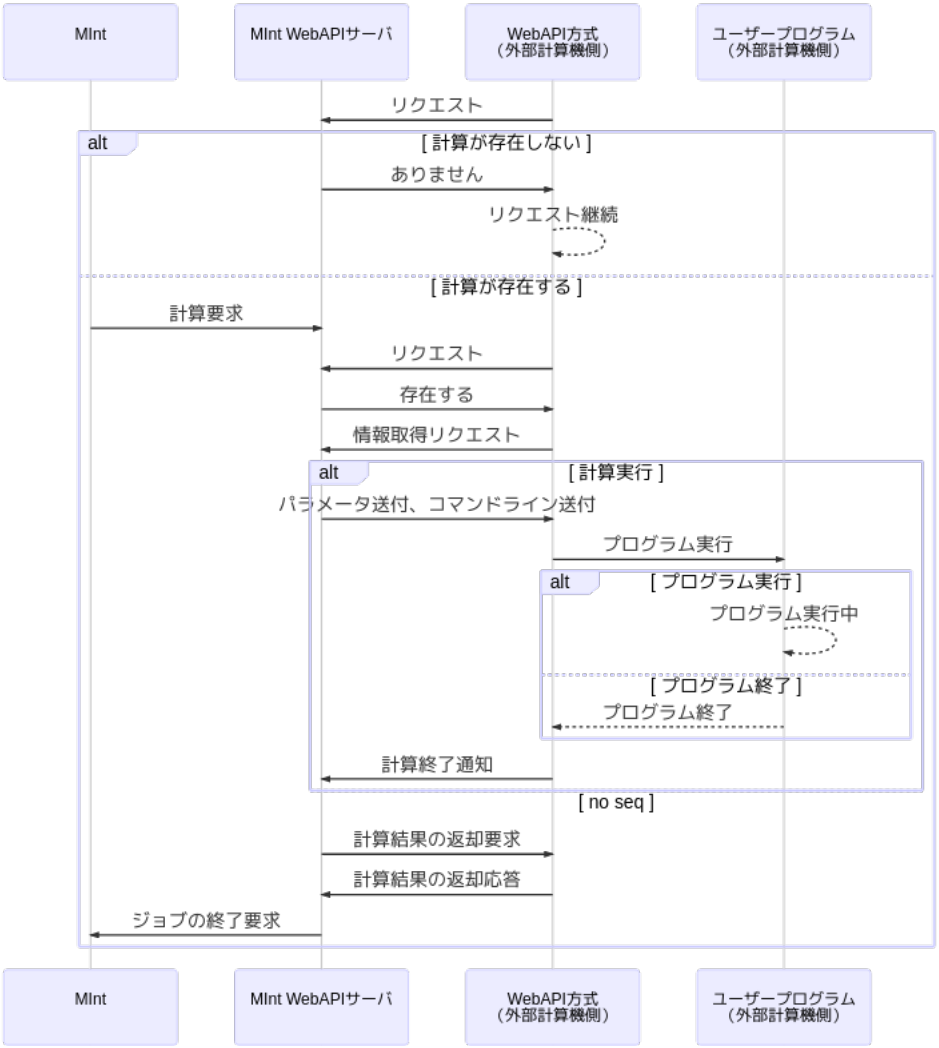

図 3.4　WebAPI 方式の流れ

3.2.2 ワークフロー例

WebAPI 方式の外部資源利用を含むワークフローを、MInt のワークフローデザイナで表示した例を示す。赤枠の部分が遠隔実行の行われるモジュールである。なお、本ワークフローは動作検証用サンプルとして、4 章の利用のための準備 で説明するインストール資材に含まれている。

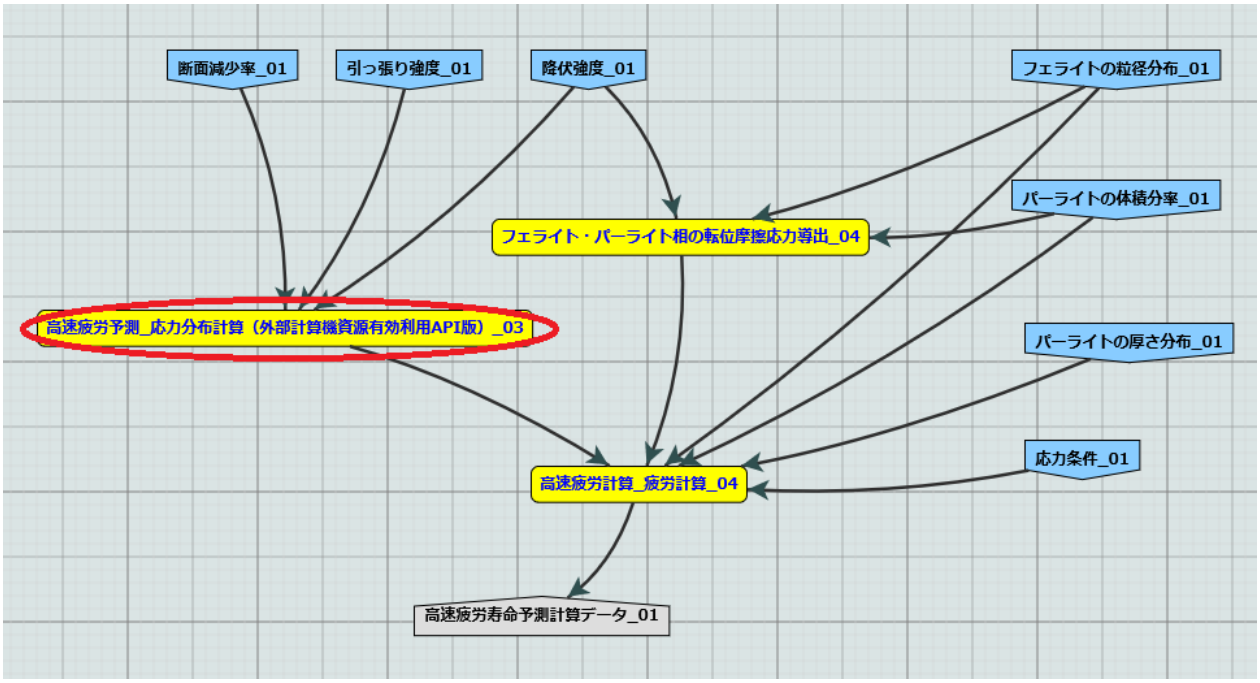

図 3.5　検証用ワークフロー

※赤枠の部分が外部計算資源を利用するモジュールである。

3.2.3 モジュール内の処理

ワークフローの当該モジュール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。

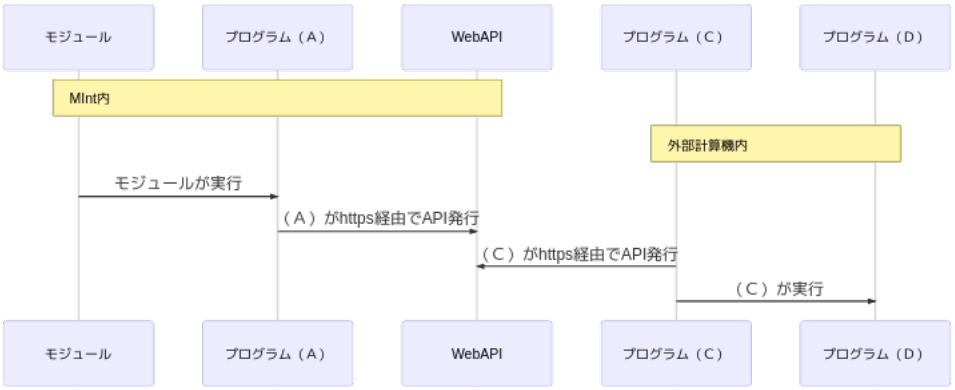

図 3.6　WebAPI 方式でのコマンドの流れ

- 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
  - 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。
  - モジュール固有の前処理を行う。
  - **misrc_distributed_computing_assist_api/debug/mi-system-side/mi-system-wf.py** を実行しておく。- 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。
  - **misrc_distributed_computing_assist_api/debug/remote-side/mi-system-remote.py** を実行しておく。
  - 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 `get_authorizaion_infomation` の get_authorizaion_infomation 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。
  - 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）

- **execute_remote_command_api.sh** を参考にして作成しておく。

第 4 章

利用のための準備

SSH 方式、WebAPI 方式それぞれのインストールおよびプログラムの実行までを外部計算機側で作業が必要な項目について説明する。なお、外部計算機側は bash スクリプトと Python スクリプトの動作する Linux 計算機を推奨している。ただし MInt 側との通信が正常に確立でき、外部計算に必要なプログラムが実行可能であれば、これ以外の環境でも構わない。その場合、こちらに無い情報は利用者側で適宜用意して使うこととなる。また、外部計算機側で秘匿データを扱う際は、これに関する仕様を MInt 側に開示する必要は無い。

4.1 事前確認

事前に以下の点を確認しておく。

- SSH 方式または WebAPI 方式の選択
- 外部計算機の利用場所
  - 自社内
  - NIMS 内
  - AWS などのクラウド

4.2 利用申請

外部計算機資源を利用したワークフローの利用申請を行い、必要な認証情報の準備を行う。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。

4.2.1 公開鍵の用意

SSH 方式を選択した場合に必要になります。SSH 接続は MInt システムから外部計算機に向けて行われます。アクセスは公開鍵暗号方式を利用したパスワード無しログインで行われます。公開鍵暗号はパスフレーズ無しの形式です。MInt 運用チームに依頼して、パスワードなしログイン用の公開鍵ファイルを入手し、以下の手順に沿ってファイルを作成しておきます。

```
$ cd .ssh
$ cat <入手した公開鍵暗号ファイル> >> authorized_keys
$ chmod 600 authorized_keys
```

注釈: .ssh ディレクトリが無い場合は作成する。パーミッションも 600 としておく。

注釈: ワークフロー実行前に MInt 運用チームに連絡してパスワードなしログインが可能なことを確認すること

4.2.2 API トークンの用意

WebAPI を選択した場合、認証情報として利用するユーザーの API トークンと識別子が必要になります。

- API トークンは 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のページで取得可能
- 識別子は文字通りユーザーを識別するもので、通常利用者の所属組織のアルファベット表記などです。

注釈: API トークンと識別子は MInt システム運用チームへもお知らせください。

4.3 環境構築

外部計算機資源を利用した計算を行う MInt システム外の計算機の環境構築について記述する。一般的に外部計算機資源を利用したいユーザーが構築する。手順は以下のとおり。

- OS の準備
- ソルバーの準備（必要であれば）
- python の準備
- git の準備
- MInt システム認証プログラム（WebAPI を選択し、API トークンを利用しない場合）
- 予測モジュール用の資材の入手（個別の利用者マニュアルを参照）
- 外部計算機資源を利用するための資材の入手と展開

4.4 OS の準備

OS は Linux 系の OS が望ましいが、利用者側で用意でき、外部計算機資源の利用が可能な OS であれば Linux に限らない。ただしその場合は利用者側が本書を読んで必要な環境を用意するものとする。

4.5 ソルバーの準備

利用したいワークフローによっては商用・非商用のソルバーが必要になる。商用の場合は利用者側でインストール、セットアップおよび動作確認を行うこととし、非商用の場合は MInt 運用チームが構築スクリプトなどを提供する。

4.6 Python の準備

利用するワークフローの予測モジュールは python で記述されたスクリプトであることがあり、これを外部計算機にインストールして利用する。対応するバージョンは 3.x であるため、用意した OS に使用可能か確認する。インストールされていない場合別途インストールする。WebAPI を選択した場合は必須となる。

4.6.1 バージョン確認

確認方法は以下のいずれかが表示されればインストールの必要はない。

```
$ python3
Python 3.x.x ～
```

```
$ python --version
Python 3.x.x ～
```

これ以外のバージョンが表示されない（コマンドが見つかりません or command not found）場合や、表示されても 2.x.x などと表示された場合はバージョン 3.x のインストールが必要です。
※ python3.x の追加インストールの方法については OS 個別となるため、利用者側のシステム管理者などにお尋ねください。

4.6.2 追加のパッケージ

pip3.x コマンドを利用して以下のコマンドをインストールする。インストールは root 権限で行う。

- requests
- urllib3

4.7 git の準備

外部計算資源の利用に必要な資材は GitHub 上のリポジトリ[*5] https://github.com/materialsintegration に用意されている。このれを取得するために git コマンドが必要となる。

[*5] 本機能を実現する資材などを格納したサーバ。GitHub を利用しているが、アカウントが無くともダウンロードは可能である。MInt 運用チームがアカウントを発行したユーザのみアップロードが可能である。

4.7.1 確認方法

git コマンドを実行してコマンドが見つかりません or command not found などになった場合は、git コマンドを別途インストールします。
※ インストールの方法については OS 個別となるため、利用者側のシステム管理者などにお尋ねください。
上記の例では以下の様な環境変数の設定が必要となります。

- MATHEMATICA11_2：Mathematica のインストールディレクトリ（例えば、/opt/Wolfram/Mathematica/11.2/Executables など）
- BRITTLE_FRACTURE_HOME：上記<展開したい場所>で指定した場所をフルパスで指定します。（例えば/home/展開したい場所/misrc_brittle/facture_workflow となる）
- QT_QPA_PLATFORM：offscreen を指定する。

4.8 外部計算機資源を利用するため資材

外部計算機に入手、展開の必要な資材は以下。

- WebAPI 用ポーリングプログラム（WebAPI 方式を選択した場合）
- MInt システム認証プログラム（WebAPI 方式の場合で、希望する場合のみ）
- SSH 用利用者側実行スクリプト集（SSH 方式の場合）

注釈: SSH 方式を選択した場合は WebAPI 方式のような特別な資材は必要ありません。

4.8.1 WebAPI 用ポーリングプログラム

WebAPI 方式を選択した場合に必要になります。手順は公開している場所から以下の要領で git コマンドを使用してダウンロード、展開します。

- 手順

```
$ cd <展開したい場所>
$ git clone https://github.com/materialsintegration/misrc_distributed_computing_
↪assist_remote_side.git
```

- 利用方法
  - proxy サーバーの設定が必要であれば、設定する。(https_proxy 環境変数）
  - 外部計算機で、外部計算を行うユーザーで以下のコマンドを実行し、WebAPI のポーリング動作を実施します。

```
$ cd <展開した場所>/misrc_distributed_computing_assist_remote_side
$ python3 mi-system-remote.py <識別子> http://nims.mintsys.jp <API トークン>
```

  - ポート番号はデフォルト 50443 を利用するが、これを利用できない場合通常の https 通信用 443 を利用することができる。以下の様に実行する。

```
$ cd <展開した場所>/misrc_distributed_computing_assist_remote_side
$ python3 mi-system-remote.py <識別子> http://nims.mintsys.jp <API トークン>␣
↪port:443
```

- 動作確認
  - 正常動作時

```
site id = 指定した識別子
base url = https://nims.mintsys.jp:50443
 token = 指定した API トークン
年/月/日 時:分:秒:send https://nims.mintsys.jp:50443/mi-distcomp-api/calc-
↪request?site_id=指定した識別子
code = 0300 / message = There is no information for accept_id(None), about␣
↪the your site id(指定した識別子)
```

  - 異常な場合その 1 ：識別子の間違い

```
code = 400 / message = Your site-id(指定した識別子) does not match in the list␣
↪that acceptable to.
```

  - 異常な場合その 2 ：API トークン間違い

```
2022/01/19 17:33:56:status code = 403 / reason = {"errors":[{"code":"0005",
↪"message":"/mi-distcomp-api/calc-request への実行権がありません。"}]}
```

- 終了方法
  - Ctrl キーと C キーを同時に押します。以下の様に表示され最大６０秒後に終了します。

```
Ctrl + C スクリプト停止要求受付
```

注釈: この作業は WebAPI 方式を選択した場合に必要です。

注釈: WebAPI を実行するユーザーでの作業となります。

4.8.2 MInt システム認証プログラム

本資材は WebAPI 方式を選択し、WebAPI ポーリングプログラムを API トークンを利用せず、通常のログイン・パスワード方式で実行する場合に必要です。

注釈: この方式ですとコマンドラインに API トークンを記述する必要がなく、安全性もたかい使用法となります。インストールは root 権限で行います。必要な場合は MInt システム運用チームまでご相談ください。

4.8.3 SSH 用利用者側実行スクリプト集

SSH 方式を選択した場合に必要になります。手順は公開している場所から以下の要領で git コマンドを使用してダウンロード、展開します。

- 手順

```
$ cd <展開したい場所>
$ git clone https://github.com/materialsintegration/remote_workflow.git
```

第 5 章

利用方法

5.1 SSH 方式

主に、外部計算機資源側の利用方法について記述する。

1. **misrc_remote_workflow** リポジトリを展開した場所の通知
   - この場所を MInt システム運用チームへ知らせる。
2. 外部計算機側で実行するスクリプトがあれば **remote-side_scripts** に配置する。
3. MInt が外部計算機へログインして最初に実行するプログラム名は前述のとおり **execute_remote-side_program_ssh.sh** に固定されている。このため **execute_remote-side_program_ssh.sample.sh** をこの名前でコピーするか、新規に作成して、必要な手順をスクリプト化する。

注釈: SSH 方式の場合は外部計算機において待機する必要のあるプログラムなどは無い。

5.1.1 (参考)MInt 側作業

1. 外部計算資源を利用するモジュールが実行可能なスクリプト **misrc_remote_workflow/scripts/execute_remote_command.sample.sh** をコピーして専用スクリプトを作成する。
2. 予測モジュールの modules/resouceRequest/pbsNodeGroup タグに ssh-node01 という値をセットする。
3. 予測モジュールの modules/objectPath タグに 1. で作成したスクリプトをセットする。
4. 1. で作成したスクリプトを各行のコメントに従い適宜修正する。
5. 1. を実行可能な予測モジュールを組み込んだワークフローを作成する。

5.2 WebAPI 方式

主に、外部計算機資源側の準備について記述する。

1. **misrc_distributed_computing_assist_api** リポジトリを展開する。
2. **authentication_operator** リポジトリを展開、環境変数を設定する。（ログイン方式を選択する場合）

注釈: 環境変数 AUTHENTICATION_OPERATOR はログインシェルの自動設定ファイルに設定しておく。

---

3. 計算に必要なスクリプトの準備
   - 独自に実行ファイルを用意した場合はの情報は MInt 運用チームに伝えておく。
   - 特別なリポジトリを利用する場合はこの作業が必要ないこともある。

5.2.1 実行

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. numref *api_polling_program* :ref: *api_polling_program* の動作確認で実行したどちからの方法で、**mi-system-remote.py** を実行する。
2. 終了する場合は Ctrl+C で停止する。
   - ポーリング中の処理がある場合に備えてすぐには終了しない。
   - 処理中の情報が無ければ最大 60 秒で終了する。

5.2.2 (参考)MInt 側作業

1. **misrc_distributed_computing_assist_api** リポジトリを展開する。
2. 構成ファイル **mi_distributed_computing_assist.ini** に必要な設定を行う。
3. **mi_dicomapi.py** を動作させて待ち受け状態にする。

```
$ python mi_dicomapi.py
```

または

```
$ systemctl start distcomp_api
```

4. モジュールの実行プログラム内で、**misrc_distributed_computing_assist_api/debug/mi-system-side/mi-system-wf.py** を必要なパラメータとともに実行するように構成する。

注釈: ワークフロー側から計算登録時に構成ファイルは再読込されるので、WebAPI プログラムが現在動作中であっても読み込ませるための特別な動作は必要ない。

5.3 外部計算機の作業場所

SSH 方式、WebAPI 方式のどちらも計算場所はワーキングディレクトリ[*6] と言い、UUID で構成されたディレクトリ名のディレクトリの作成時間などで該当ディレクトリかどうか判断する。

5.4 その他 MInt 側注意事項

SSH 方式、WebAPI 方式共通の注意事項など。

- pbsNodeGroup 設定で ssh-node01 を設定する。他の計算機では外へアクセスすることができないため。
- pbsQueue など CPU 数などは指定できない。
- 外部計算機側で別途 Torque などのバッチジョブシステムに依存する。

5.5 エラーが発生した場合

ワークフローを本実行する前に MInt 運用チームと連携して動作確認を行っておくが、予期せず異常終了した場合などは以下の方法で対策を検討することができる。

- SSH 方式、WebAPI 方式ともワークフローの出力ポートとは別に外部計算機で計算が行われた際の処理のログがある。MInt 運用チームに連絡して、それを入手する。
- WebAPI 方式であれば通信ログがポーリングプログラムの実行画面に出力されるのでこれを利用する。
- 同様に、外部計算機のワーキングディレクトリに計算結果およびログが残っているのでこれを利用する。

5.5.1 通信異常

インターネット経由であるので、通信異常は発生するものとして対処してある。WebAPI 方式では外部計算機側からの通信となるため、外部計算機側のプログラムでリトライ方式を採用している。デフォルトはリトライ間隔 60 秒のリトライ回数 5 回で通信失敗として終了する。この値は以下の書式で上書き指定することも可能である。

```
$ python mi-system-remote.py <ホスト情報> https://nims.mintsys.jp <API token> retry:<リトライ回数>,<リトライ間隔>
```

SSH 方式ではその性質上処理中に通信異常が起きると復帰できない。現バージョンでは SSH 方式での処理中の通信異常を復帰させる手段は実装されていない。どちらの場合も通信が途絶えて処理続行不能と判断されれば、MInt システム側に異常を通知し、異常終了となる ように構成されている。

注釈: リトライ回数は整数で指定し、リトライ間隔は整数または実数で指定する。

[*6] 外部計算機側のワーキングディレクトリは/tmp ディレクトリに作成されるので、OS の設定に変更がなければ 30 日後に削除される。このため問題が発生した場合は発生から 30 日以内に調査を開始する必要がある。

5.6 ワークフローの廃止

ユーザが MInt 運用チームにワークフローの廃止届を提出する。当該ワークフローは MInt 上で「無効」のステータスを付与され参照・実行不能となる。

以上