外部計算資源の利用について

リリース ***0.2.0***

SIP-MI

2021 年 **06** 月 **22** 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。

- 部外秘プログラムの使用
- 部外秘データの使用
- 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. 産学共同研究契約、MInt システム利用規定 (★正式名称調べる by 酒井さん★) 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 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るため、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。

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。

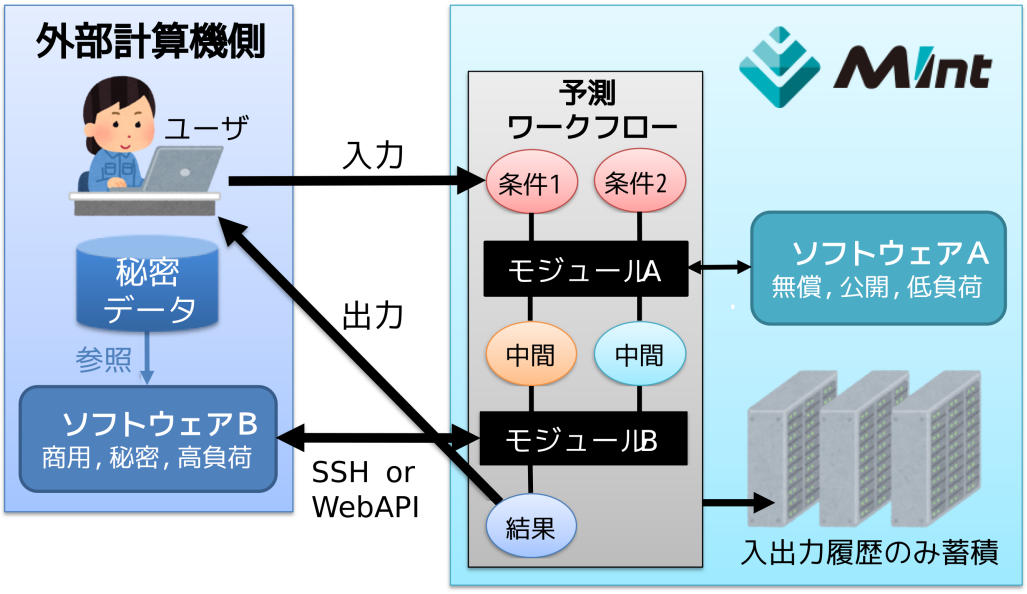

図 2.1　外部計算資源の利用イメージ

[*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。
- WebAPI 方式
  - 外部計算機から MInt の API 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 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 API サーバへの https(TCP/443) のアウトバウンドアクセスの許可が必要である。

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。

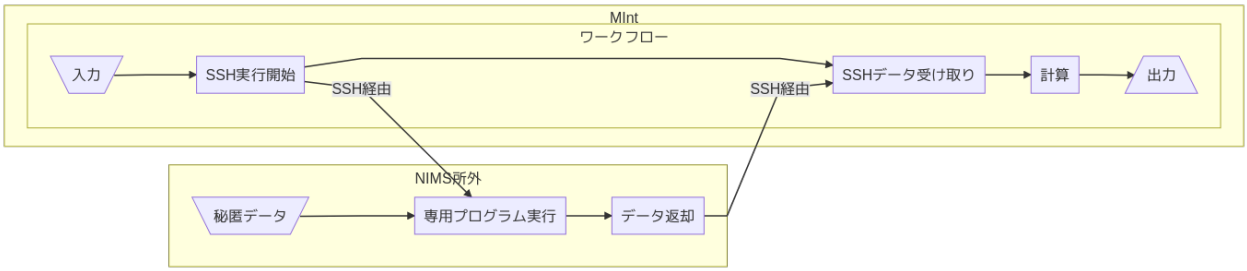

図 3.1　SSH 方式の外部資源利用のイメージ

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、使用方法 で説明するインストール資材に含まれている。

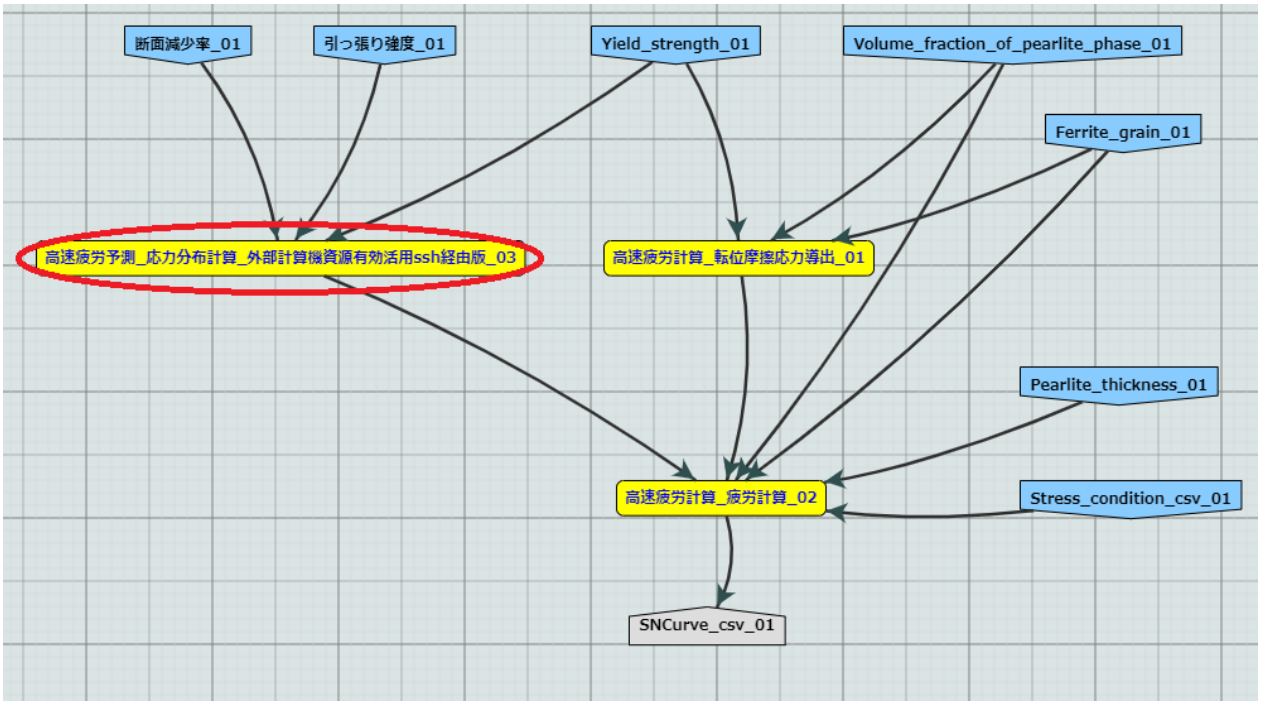

図 3.2　動作検証用のワークフロー

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。

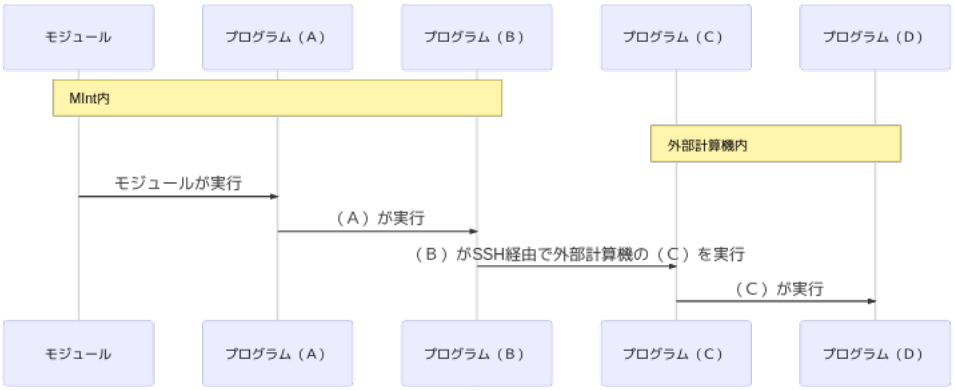

図 3.3　SSH 接続経由によるコマンド実行の流れ

- 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。
  - 使用方法 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「execute_remote_command.sh」固定である。(インストール資材に含まれるサンプルファイルはリネームが必要)
  - 使用方法 で説明する編集を行う。
  - 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[*2] 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「execute_remote-side_program_ssh.sh」固定である。(インストール資材に含まれるサンプルファイルはリネームが必要)
  - 外部計算機上で実行するプログラムは、ここへシェルスクリプトとして記述する。
- プログラム（D）
  - 必要に応じて（C）から実行される外部計算用スクリプト群。

[*2] 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。

- 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。

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。

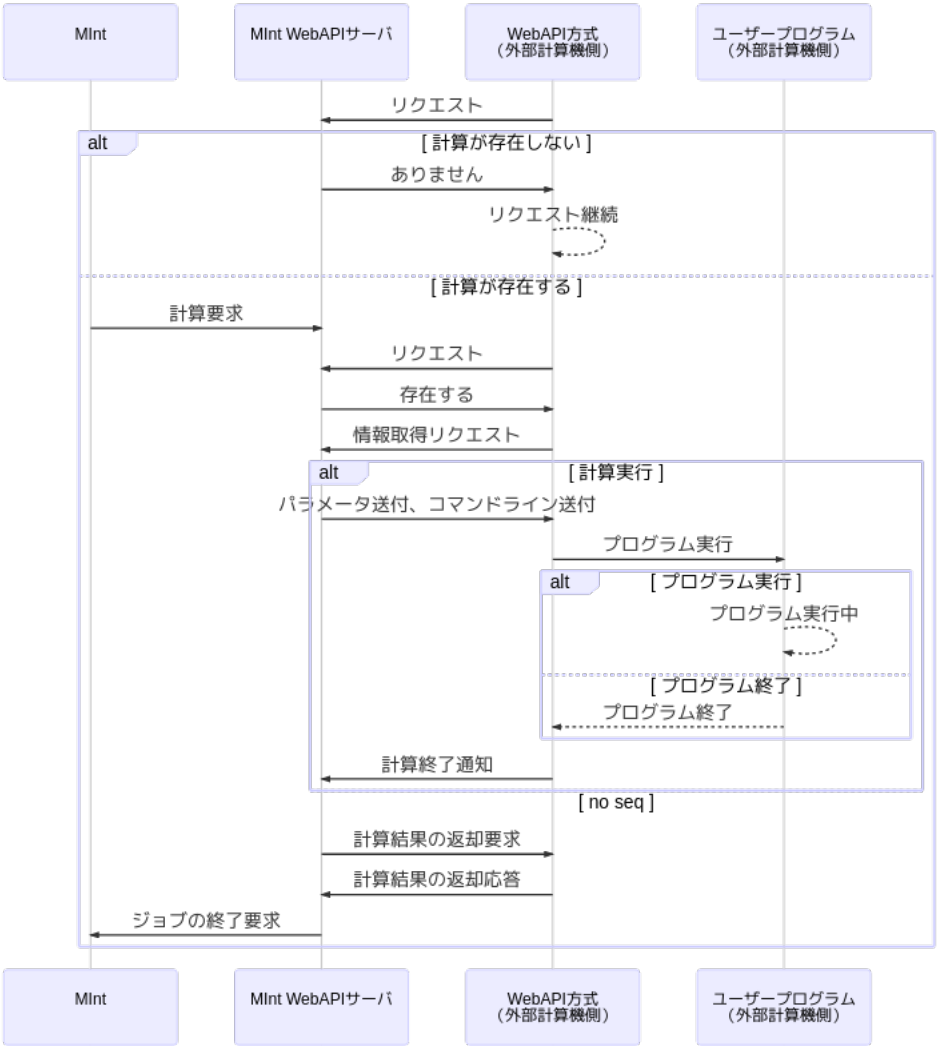

図 3.4　WebAPI 方式の流れ

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、使用方法 で説明するインストール資材に含まれている。

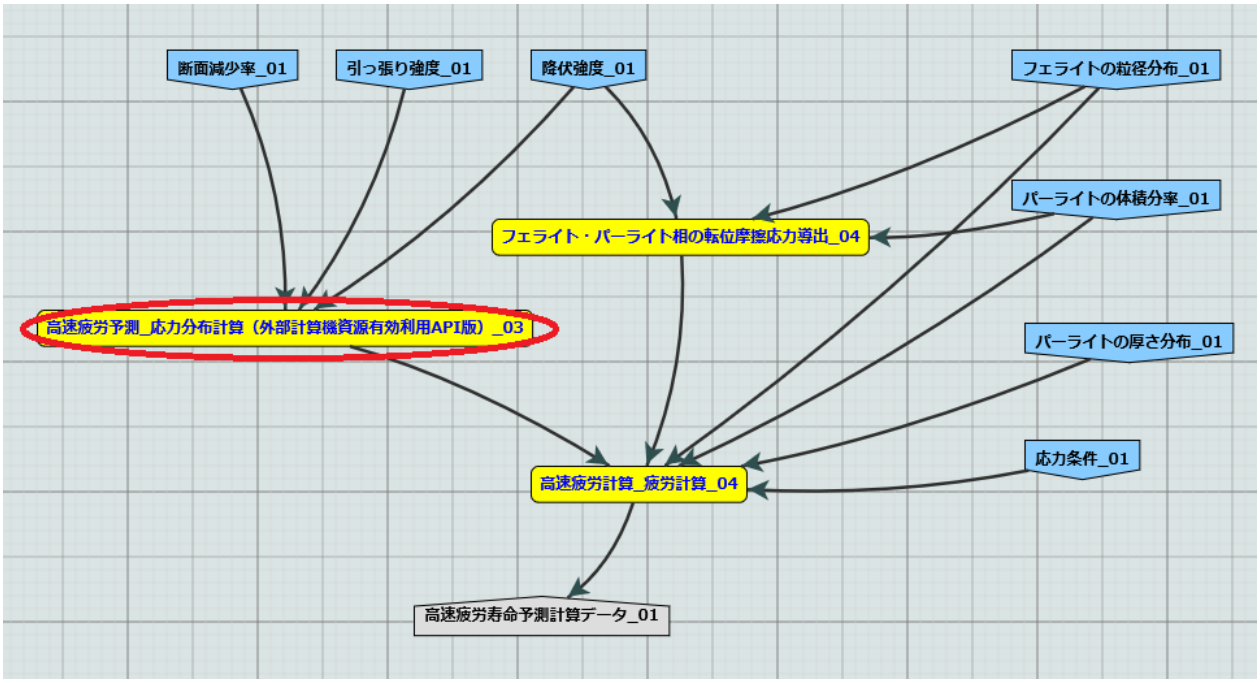

図 3.5　検証用ワークフロー

※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。

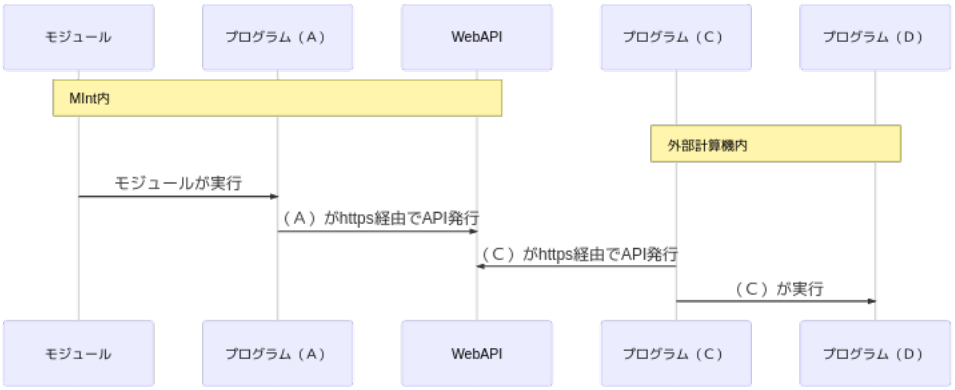

図 3.6　WebAPI 方式でのコマンドの流れ

- 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。
  - misrc_distributed_computing_assist_api/debug/mi-system-side/mi-system-wf.py を実行しておく。
  - WebAPI へ計算の情報を登録する。
- 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。
- 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。
- 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

使用方法

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説明する。なお、外部計算機は bash スクリプトと Python スクリプトの動作する Linux ホストを想定しているが、MInt 側の通信が正常に確立できるならば、これ以外の環境でも構わない。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 側に開示する必要も無い。

4.1 事前決定事項

事前に決定しておく項目は以下の通り。

1. 環境構築
   - 外部計算機側, MInt 側のユーザアカウントの準備
   - SSH or WebAPI の方式選択
   - 認証関連情報の準備
2. ワークフロー・モジュールの仕様策定 (実装調査書の作成)
   - MInt と外部計算機の役割分担の決定
   - MInt と外部計算機の間を受け渡すパラメータ・ファイルの設計
   - MInt 側の前処理・後処理の設計
   - 外部計算機側スクリプトの設計

4.2 SSH, WebAPI 方式共通

4.2.1 資材の入手

外部計算資源の利用に必要な資材は GitHub 上のリポジトリ[*4] https://github.com/materialsintegration に用意されている。

- misrc_remote_workflow
  - 主に外部計算機側で実行されるスクリプトのサンプルが同梱されている。
- misrc_distributed_computing_assist.api
  - WebAPI 方式用のプログラムおよびサンプルが同梱されている。

[*4] 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、MInt 運用チームがアカウントを発行したユーザのみダウンロードが可能である。

  - MInt 側資材は「debug/mi-system-side」、外部計算機側資材は「debug/remote-side」にある。

ユーザは外部計算機上にこれらを展開し、必要なカスタマイズを行う。資材展開後の外部計算機側のディレクトリ構造は以下のようになる。

- ユーザーディレクトリ

```
~/ユーザーディレクトリ
  + remote_workflow
    + scripts
      + input_data
  + misrc_distributed_computing_assist_api
    + debug
      + remote-side
```

- ワーキングディレクトリ

```
/tmp/<uuid>
```

ユーザが外部計算機側でカスタマイズするファイルは、通常、SSH 方式では「misrc_remote_workflow/scripts/execute_remote-side_program_ssh.sample.sh」、WebAPI 方式では「execute_remote-side_program_api.sample.sh」のみである。カスタマイズの方法については後述する。これ以外のファイルも改変可能であるが、その改変が原因で外部計算を利用するワークフローが動作しなかった場合、MInt 運用チームは責を負わない。

資材展開後の MInt 側のディレクトリ構造は以下のようになる。

- ユーザーディレクトリ

```
~/misystem ディレクトリ
 + remote_workflow
   + scripts
 + misrc_distributed_computing_assist_api
   + debug
     + mi-system-side
```

- ワーキングディレクトリ
  - 複雑なので省略する。

4.2.2 ワークフローサンプル

misrc_remote_workflow/sample_data に、Abaqus 実行環境が用意可能な場合に使用可能なワークフローおよび、そのサンプル入力ファイルが用意されている。これを利用して、MInt 側と外部計算機側のテストが可能である。また、misrc_remote_workflow/scripts に、この時のモジュール実行プログラムがある。これを参考に、他のモジュール実行プログラムを作成することが可能である。利用する際には MInt システム側と調整が必要である。

- kousoku_abaqus_ssh_version2.py : SSH 方式のモジュール実行スクリプト
- kousoku_abaqus_api_version2.py : WebAPI 方式のモジュール実行スクリプト

もっと簡易な例題ワークフローは現在準備中である。

4.3 SSH 方式

4.3.1 公開鍵の用意

パスフレーズ無しの公開鍵認証を原則とする。外部計算機側で作成した RSA 公開鍵 (例: ~/.ssh/id_rsa.pub) を MInt 運用チームに送付する。鍵は既存のものでも良いが、下記のコマンドで新規に作成しても良い。

```
$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/misystem/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/misystem/.ssh/id_test_rsa.
Your public key has been saved in /home/misystem/.ssh/id_test_rsa.pub.
The key fingerprint is:
fd:f6:ab:3c:55:8d:f5:4d:52:60:27:2b:9b:b8:49:fb misystem@zabbix-server
The key's randomart image is:
+--[ RSA 2048]----+
|               +oo|
|              ..+o|
|             . .=+|
|          . . +. =|
|         S + o  . |
|          . =  .  |
|           + o.   |
|            +..   |
|             Eoo. |
+-----------------+
```

4.3.2 資材の展開

1. misrc_remote_workflow リポジトリを展開する。

```
$ git clone https://gitlab.mintsys.jp/midev/misrc_remote_workflow
$ cd misrc_remote_workflow
$ ls
README.md  documents  inventories  misrc_remote_workflow.json  modulesxml  sample_
↪data  scripts
$ cd scripts
$ ls
abaqus                                    execute_remote_command.sample.sh ␣
↪kousoku_abaqus_ssh.sh
create_inputdata.py                       input_data                        ␣
↪kousoku_abaqus_ssh_version2.py
execute_remote-side_program_api.sample.sh  kousoku_abaqus_api_version2.py   ␣
↪kousoku_abaqus_ssh_version2.sh
execute_remote-side_program_ssh.sample.sh  kousoku_abaqus_api_version2.sh   ␣
↪remote-side_scripts
```

(次のページに続く)

(前のページからの続き)

```
execute_remote_command.sample.py          kousoku_abaqus_http.py
```

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。

4.3.3 (参考)MInt 側作業

1. 外部計算資源を利用するモジュールが「misrc_remote_workflow/scripts/execute_remote_command.sample.sh」に相当するスクリプト (実際にはリネームされている) が必要なパラメータとともに実行するように構成する。
2. 1. を実行可能なワークフローを、外部計算を含まないものと同じ手順で作成する。

4.4 WebAPI 方式

4.4.1 認証関連情報の用意

MInt 側担当者に問い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。ユーザ企業のドメインなどと一致させる必要は無い。
- 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。MInt 運用チームに問い合わせて取得する。
- 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。

4.4.2 資材の展開

1. misrc_distributed_computing_assist_api リポジトリを展開する。

```
$ git clone https://gitlab.mintsys.jp/midev/misrc_distributed_computing_assist_api
$ cd misrc_distributed_computing_assist_api
$ ls
README.md  logging.cfg     mi_dicomapi_infomations.py           syslogs
debug      mi_dicomapi.py  mi_distributed_computing_assist.ini
$ cd debug
$ ls
api_status.py  api_status_gui.py  api_status_gui.pyc  mi-system-side  remote-side
$ cd remote-side
$ ls
api-debug.py  debug_gui.py  mi-system-remote.py
```

2. mi-system-remote.py を実行する

```
$ python mi-system-remote.py <ホスト情報> https://nims.mintsys.jp <API token>
```

4.4.3 (参考)MInt 側作業

1. misrc_distributed_computing_assist_api リポジトリを展開する。
2. mi_dicomapi.py が未動作であれば、mi_distributed_computing_assist.ini に外部計算機の設定を実施する。動作中であれば、設定を再読み込みする。

```
$ python
>>> import requests
>>> session = requests.Session()
>>> ret = session.post("https://nims.mintsys.jp/reload-ini")
>>>
```

3. mi_dicomapi.py を動作させて待ち受け状態にする。

```
$ python mi_dicomapi.py
```

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。

4.4.4 その他 MInt 側注意事項

- 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。

4.5 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