
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 11 日

目次:

第 1 章	概要	1
第 2 章	初めに	2
2.1	特徴	3
2.2	扱う方式	4
2.3	使用するリポジトリ	4
2.3.1	展開したファイルの扱い	4
2.4	諸条件	5
2.4.1	SSH 方式でのアクセスのために	5
2.4.2	WebAPI 方式でのアクセスのために	5
2.5	実行されるコマンド	5
2.6	送受信されるファイルについて	5
2.6.1	特殊な実行を行う場合	6
2.6.2	データの帰属に関して	6
2.7	実行ユーザーについて	6
2.8	使用を終了した後	6
第 3 章	外部計算機資源を MInt システムから有効に活用するための手法とは	7
3.1	SSH を利用した遠隔実行	7
3.1.1	概要	7
3.1.2	実行のイメージ	8
3.1.3	システム要件概要	9
3.1.4	MInt システム側	9
3.1.5	外部計算機資源側	9
3.1.6	用意されているサンプルワークフロー	10
3.1.7	外部計算機でのディレクトリ	11
3.1.8	コマンドの流れ	11
3.1.9	MInt システムと送受信されるデータ	12
3.2	WebAPI 方式	13
3.2.1	概要	13
3.2.2	実行のイメージ	14
3.2.3	WebAPI 方式の流れ	15
3.2.4	システム要件	16

3.2.5	用意されているサンプルワークフロー	17
3.2.6	MInt システムでのディレクトリ	18
3.2.7	外部計算機資源でのディレクトリ	18
3.2.8	外部計算機で MInt システムから実行されるプログラム	19
3.2.9	MInt システムで送受信されるデータ	19
第 4 章	使用方法	20
4.1	使用開始前に	20
4.1.1	事前決定事項の列挙	20
4.1.2	SSH 方式の認証情報	21
4.1.3	WebAPI 方式の認証情報	21
4.1.4	外部計算機資源側の準備	21
4.1.5	MInt システム側の準備	21
4.1.6	専用予測モジュールの準備	21
4.1.7	WebAPI 方式の場合の設定事項	22
4.2	SSH 方式	23
4.2.1	外部計算機資源側	23
4.2.2	MInt システム側	23
4.3	WebAPI 方式	23
4.3.1	外部計算機資源側	23
4.3.2	MInt システム側	24
4.4	ワークフローについて	24
4.4.1	共通事項	24
4.4.2	SSH 方式	25
4.4.3	WebAPI 方式	25
4.4.4	サンプル	25
参考文献		27

第 1 章

概要

MInt には、ワークフローを構成する各モジュール中の任意の部分を MInt の計算ノード以外に処理させることができる「外部計算資源利用」機能がある。ユーザはこの機能を利用してローカルで処理を行わせることにより、下記のような利点を得られる。

- ローカルの部外秘プログラムを使用できる
- ローカルの部外秘データにアクセスできる
- ローカルの特殊 (MInt の計算ノードでは対応できない) 構成の計算機を使用できる
- ローカルの商用ソフトを使用できる (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、ライセンス規定により、ほとんどの場合 NIMS 所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 システム、外部システムの双方が必要なセキュリティ水準 (後述) を満たしている必要がある。また、両者間のネットワークは常時 SSL で暗号化されている。

外部計算資源には、SSH と WebAPI というふたつの方式がある。前者は MInt から外部計算機へ SSH でアクセスし、必要なデータとコマンドをプッシュする方式である。単純明快であり、外部処理を遅延なく開始できるという利点があるが、外部システム側で MInt に対し SSH アクセスを認める (ポート開放する) 必要がある点は、特に企業ではハードルが高いと想定される。一方、後者は数分程度の間隔で外部システム側から MInt に WebAPI 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 API への応答としてプルされてくる方式である。この方式では外部システム側にポート開放の必要が無いが、外部処理の開始までに最大でポーリング間隔分の遅延が生じる。

MInt でシステムが収集する情報はワークフローの各ラン (run) におけるモジュールの入口と出口の情報のみであり、モジュールの内部で一部処理を「外注」する本機能は情報収集の対象外である。SSH や WebAPI の通信内容が収集されることはない。また、SSH でもっとも広く利用されている OpenSSH には、標準で (.authorized_keys の設定で) ユーザが MInt に実行させるコマンドを固定する機能がある。さらに、ユーザはローカルから送出されるデータを自らの裁量で十分限定することができる。

これらの仕組みによって、ユーザは安全に外部計算資源利用機能を活用することができる。下記の各章で、必要なセキュリティ条件、ならびに具体的な実装方法について記す。

第 2 章

初めに

外部計算資源を利用するにあたっては、下記の点に留意する。

1. 産学共同研究契約、MInt システム利用規定に遵守する。
2. MInt システムにおいては、セキュリティ上の観点で下記の点について対策を実施している。
 - 第三者によるセキュリティ分析・セキュリティリスク診断及び対策を実施している。また、定期的に脆弱性情報を確認し、対策を実施している。
 - 提供するツールについても脆弱性診断を実施している。
 - アクセス監視やネットワーク負荷監視を実施している。
3. 外部計算資源の利用においては、
 - 外部計算資源において十分なセキュリティ対策を実施すること。また、継続的に利用される場合には、定期的な脆弱性診断を実施し対策を講じる。
 - SSH による外部計算資源を利用する場合には、外部計算資源側の受け入れとして、接続元 IP のみ接続可能とする設定を行う。
4. 不明な点は、お互い協議をして決定する。

2.1 特徴

本システムの特徴を以下に挙げる。

- NIMS DMZ に配置した MInt システムの機能の一部である
 - SSH 等で外部の計算機を外部計算機資源として活用可能。
 - 専用 API システムを利用し、SSH 接続が不可能な外部計算機資源を活用可能。
- 外部計算機資源における秘匿データを扱うことが可能である
 - 秘匿データの指定は外部計算機資源側で設定され隠蔽可能。
 - MInt システム側からはその存在は感知できない。
- 独自アプリケーションの利用を想定
 - 外部からアクセスした場合に MInt システムにアクセスした場合に利用できないアプリケーションの利用を想定。
 - アプリケーションの実行設定も外部計算機資源側で行われる。
 - MInt システムは計算結果だけを送付されるのでアプリケーションの存在は感知できない。

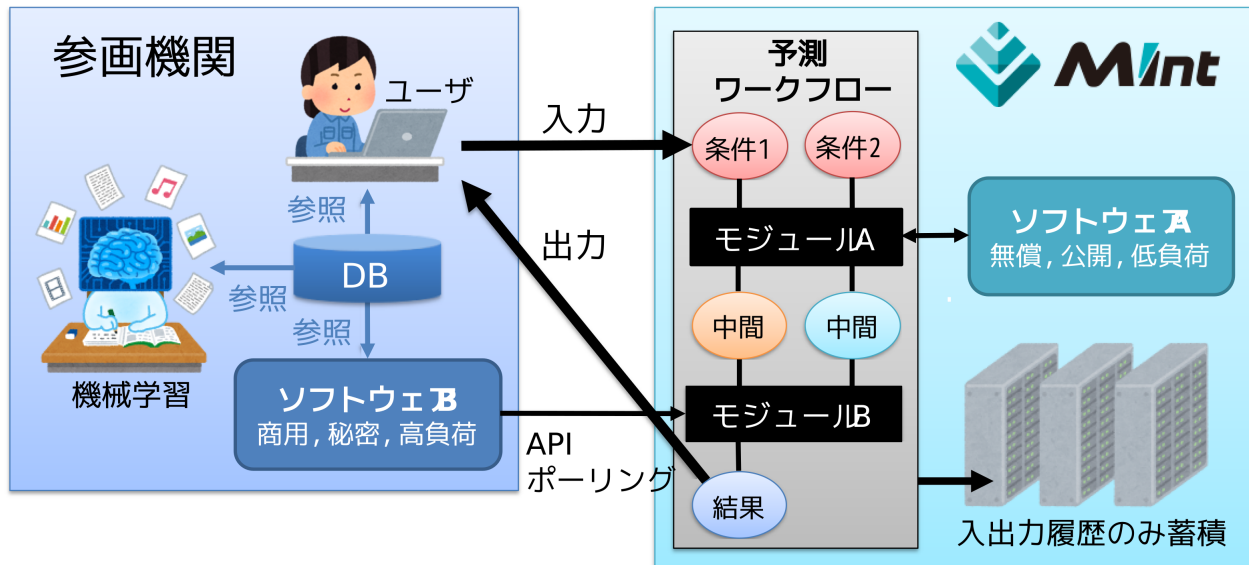


図 2.1 外部計算機資源利用機能を活用した計算のデータの流れ

本ドキュメントは外部計算機資源の有効活用について、動作原理などを説明し、次いで簡単にインストール、実行する方法を説明する。インストール、実行は MInt システム側と外部計算機資源側に分かれています。

NIMS の取り組みについてはこちら [\[activities_of_NIMS\]](#) を参照。

2.2 扱う方式

本書で扱う外部計算機資源の有効活用の方法は以下の2つの形式である。

- SSH 方式
 - SSH 機能を利用して MInt システムの任意のモジュールから直接遠隔計算を実行する。すべての通信は SSH 機能を利用したリモートコマンド実行により行われる。
- WebAPI 方式
 - MInt システム、外部計算機資源の中間に位置する API サーバーを構築。これを利用して MInt システムの任意のモジュールの計算を外部計算機資源側からポーリングすることで遠隔計算を実行する。全ての通信はこの API サーバーを経由して行われる。

2.3 使用するリポジトリ

外部計算機資源の有効利用のために、以下2つのリポジトリ^{*1}を用意してある。外部計算機資源側はこれらを外部資源計算用計算機に配置し、プログラム実行に必要なコマンド、ファイル送受信の手続きを設定、埋め込むだけでよい。

- misc_remote_workflow
 - 主に外部計算機資源側で実行されるスクリプトのサンプルが登録されている。
- misc_distributed_computing_assist.api
 - WebAPI 方式のためのシステム構築用のプログラム、サンプルが登録されている。
 - ワークフローで使用するプログラムは「debug/mi-system-side」にある。
 - 外部計算機側で使用するプログラムは「debug/remote-side」にある。

2.3.1 展開したファイルの扱い

リポジトリから外部計算機資源の有効利用のためにダウンロードしたファイル類には以下の制約を果すものとする。

1. 外部計算機資源としての実行に関する設定等を載せる一部のファイル^{*2}を除いてライセンスは「」が適用される。
2. 1. の一部のファイルを除くソースコードの著作権は MInt システムが保持する。
3. 外部計算機資源側での独自の改変は自由とするが、それによって外部計算機資源の有効利用のワークフローが動作しなくなっても MInt システム側は責任を追わないこととする。
4. リポジトリのアクセスはダウンロードのみとし、外部計算機資源側での独自の改変は反映されない。
5. 外部計算機資源側独自の改変を 1. 以外のスクリプトに適用したい場合は MInt システムとの個別相談となる。

^{*1} 本機能を実現する資材などを格納したサーバーのこと。GitHub を利用する。格納場所は MInt システムが用意する。アカウント制御されており、限られたアカウントのみダウンロード可能。アップロードは MInt システムが許可したアカウントのみ可能である。クラウドサーバーの様な使い方が可能であり、ネット経由で必要なファイル（ソースコードや各種ドキュメント）をダウンロード可能なので USB メモリや CD-ROM などの物理メディアに頼る必要が無い。

^{*2} misc_remote_workflow/scripts 以下にある、SSH 方式を選択した場合の execute_remote-side_program_ssh.sample.sh を複製したファイルと WebAPI 方式を選択した場合の execute_remote-side_program_api.sample.sh 及びこれらを複製したスクリプトファイルを指す。

2.4 諸条件

各方式の最低限の条件を以下に挙げる。各方式の使用開始前に必要な設定事項は後述 ([使用方法](#) を参照) する。

2.4.1 SSH 方式でのアクセスのために

ワークフローから SSH コマンドを利用して、外部計算機資源にアクセスするために計算機を設置する企業または機関には SSH 接続が可能な処置が必要である。

- SSH プロトコルの使用許可および使用ポートの開放。

2.4.2 WebAPI 方式でのアクセスのために

SSH での利用が不可能な場合、本方式を使用する。本方式ではワークフロー側は直接外部計算機資源にアクセスせず、外部計算機資源側で定期的に問い合わせ（ポーリング）する必要がある。ポーリングには通常の https 通信を用いる。このための処置が必要である。

- https プロトコルの使用許可および使用ポートの開放。

2.5 実行されるコマンド

SSH 方式の場合、リポジトリにあるコマンドしか外部計算機上では実行しない。その場所も事前に取り決めた場所となる。WebAPI 方式も同様である。

2.6 送受信されるファイルについて

- SSH 方式で送受信される、ファイル、文字列の形式
 - 暗号化 SSH プロトコル使用
 - ファイル
 - * 非圧縮 (rsync コマンドを a オプション v オプションと共に使用)
 - * ファイルサイズ無制限
 - コマンド列などの文字列
 - * 圧縮も base64 エンコードもしない平文
- WebAPI 方式で送受信される、ファイル、文字列の形式
 - 暗号化 SSH プロトコル使用
 - ファイル
 - * 内容は base64 エンコード
 - * 非圧縮
 - * ファイルサイズ最大 2G バイトまで
 - コマンド列などの文字列
 - * 圧縮も base64 エンコードもしない平文
- どちらの方式もワークフローの入出力ポートのファイル以外の通信内容を保存、保持はしない。

2.6.1 特殊な実行を行う場合

外部計算機資源での計算の前段または後段、その両方において MInt システム上で計算を行うことが可能であるが、この時送り込まれる該当予測モジュールに関係ないいかなるデータ、ファイルも MInt システムは感知しないし、GPDB などへの登録もしないが、ストレージには残る。必要であれば関連スクリプト中で最後に削除の手続きを実施することを推奨する。

2.6.2 データの帰属に関して

2.7 実行ユーザーについて

外部計算機資源側、MInt システム側双方でアカウントが必要となる。外部計算機資源側はそれぞれの組織内で決定し、MInt システム側はこの機能を利用する組織が MInt システム側へアカウント作成の依頼を行う。双方またはどちらかにすでにアカウントがある場合はそれを利用することになる。

2.8 使用を終了した後

本機能を利用したワークフローの使用を永久に停止する場合は、利用を開始したときと同様に廃止届けを提出する。受理されると MInt システムでは該当ワークフローは「無効」ステータスとなり実行できなくなる。また該当予測モデルは権限が MI システム管理者のみとなり使用可能なユーザーがほぼ 0 となる。

第 3 章

外部計算機資源を MInt システムから有効に活用するための手法とは

最初に各手法の動作原理を説明する。

3.1 SSH を利用した遠隔実行

最初に SSH を利用して、MInt システムの任意のモジュールから外部計算機資源を利用する方法を説明する。

3.1.1 概要

SSH を利用した遠隔実行とは、SSH プロトコルを利用してネット上でアクセス可能な場所にある計算機をあたかも MInt システムの計算機の一部として使用すること言う。この場合 SSH パケットが到達可能な場所であればどこでも対象となり得る。SSH アクセスではパスワードなしでの運用も可能であり、本システムも基本的にパスワードなし接続での運用が前提であるが、必ずしも必須ではない。スクリプト内で実現可能であればパスフレーズ付きなど多彩なアクセス方法を採用可能となっている。

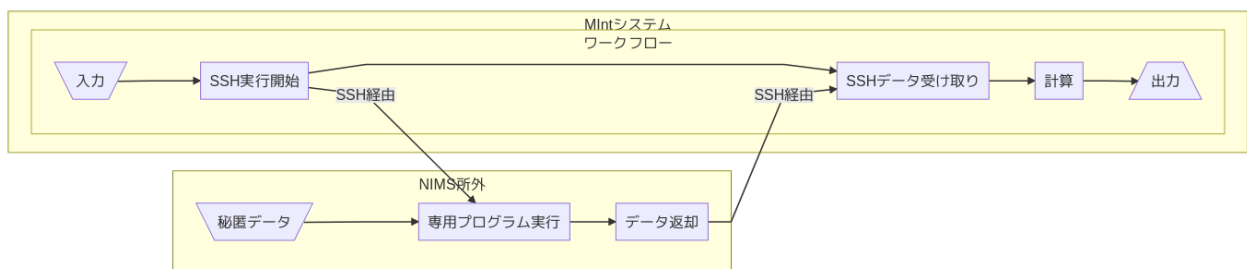


図 3.1 SSH 実行のイメージ

3.1.2 実行のイメージ

この方式では、以下のようなシステムで動作サンプルが用意されている。

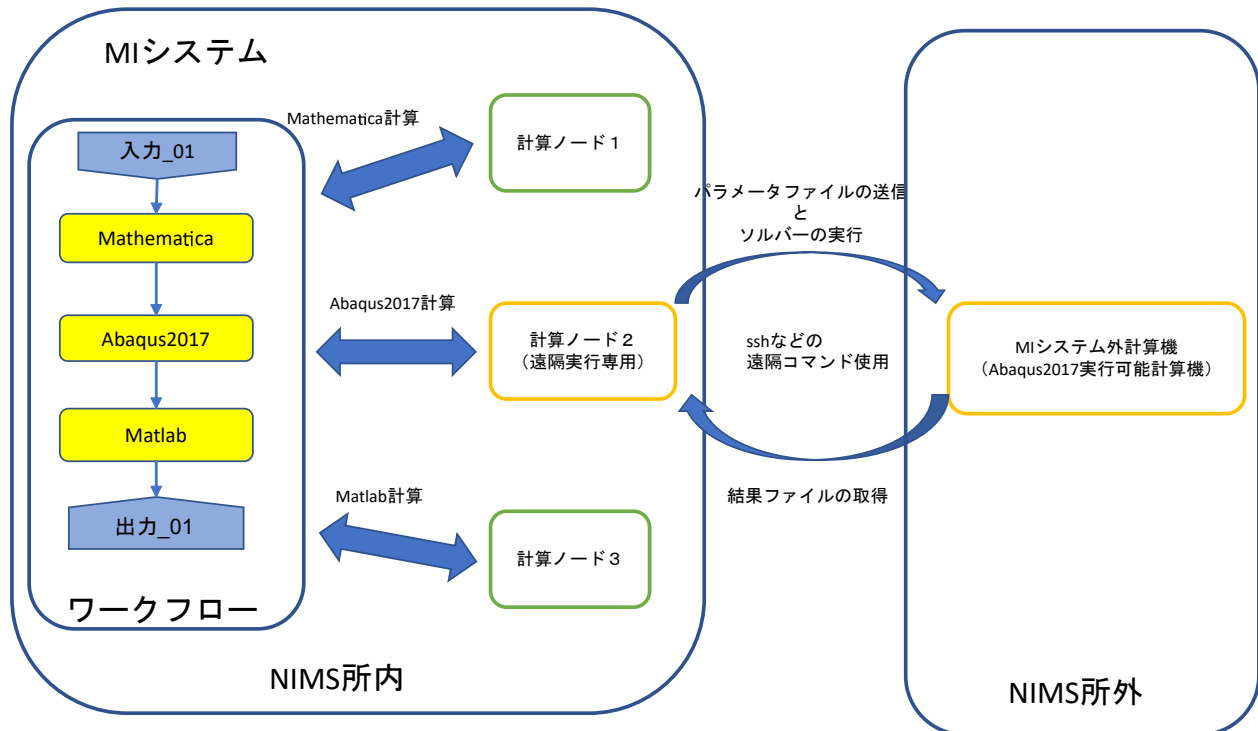


図 3.2 遠隔実行のイメージ

このようにして、特定のモジュール（Abaqus2017）と特定の計算ノード（計算ノード2）を用意し、計算ノード2がMIntシステム外にある計算機を遠隔実行できるように設定して、使用することでMIntシステム外の計算機または計算機群をMIntシステム内にあるかのごとく計算（ワークフロー）を実行することが可能になる。またAbaqus2017と謳ってはいるが実行するプログラムはこれに限らず、様々なコマンド、プログラム、アプリケーションを実行することが可能なように作られている。

3.1.3 システム要件概要

- MInt システム側
 - 遠隔実行専用の計算ノードを設置してある。
 - 遠隔実行用予測モジュールを作成。
 - このモジュールは専用計算ノードを指定して計算を行うよう設計。
 - モジュールおよび専用計算ノードに SSH 操作の設定。
- 外部計算機資源側
 - 外部から到達可能な場所。
 - Linux 計算機を想定する。(Mac でも可能。Windows は SSH 到達に問題があるため非推奨)。
 - 必要な資材を取得、展開。
 - 必要な情報を設定。(主に実行プログラムパス、パラメータ、秘匿データの配置)

3.1.4 MInt システム側

専用計算ノードでは以下のような動作が行われるように、専用モジュールが定義するプログラムを実行する。
必要な資材は GitHub に登録してある。

- パラメータ類の遠隔計算機へ送信（遠隔計算機側にあるパラメータまたはファイルを指定することも可）。
- 遠隔計算機でソルバー（プログラム）の実行。
- 実行が終了したら結果ファイルの取得。

3.1.5 外部計算機資源側

外部計算機資源側計算機では、必要なファイルの配置が主な手順である。
必要な資材は GitHub に登録してある。

- 資材の展開
- 実行プログラムパスの調整
- 秘匿データ（ある場合）の指定ディレクトリへの配置

3.1.6 用意されているサンプルワークフロー

サンプルとして下記のようなイメージの動作検証用ワークフローを用意してある。

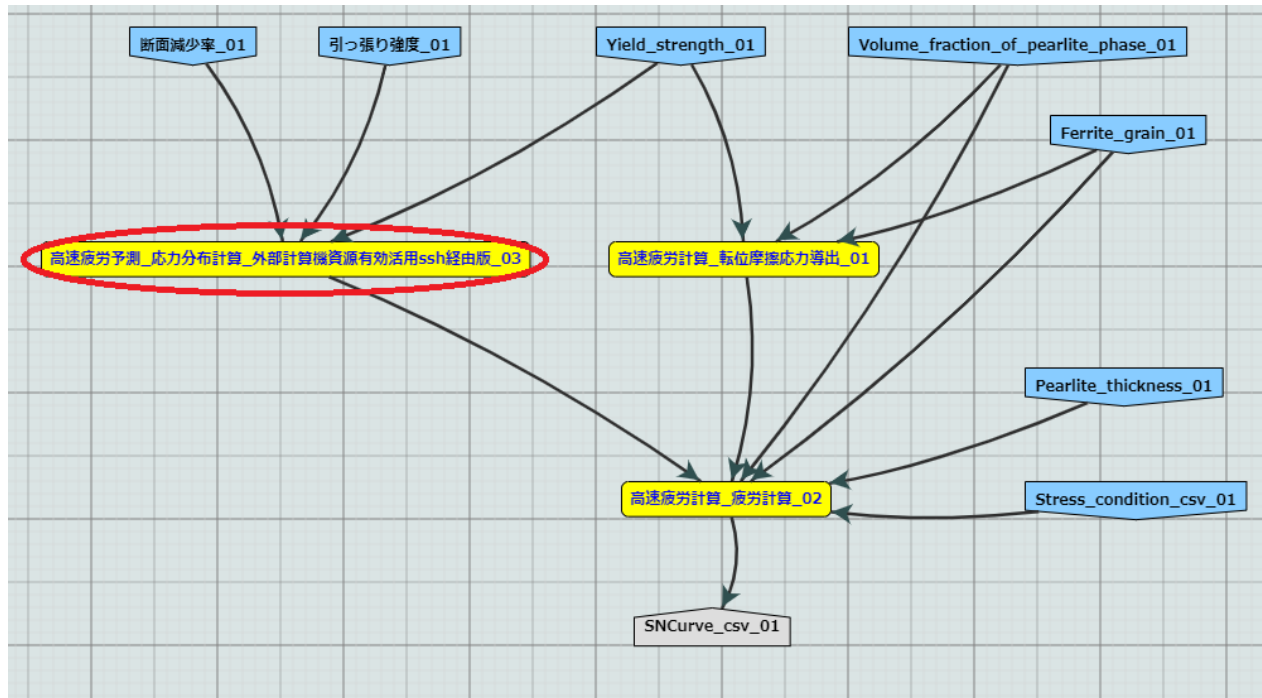


図 3.3 動作検証用のワークフロー

※赤枠の部分が遠隔実行の行われるモジュールである。

3.1.7 外部計算機でのディレクトリ

外部計算機のディレクトリ構造は以下のようになっている。インストール方法については後述する。

- 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
```

- ワーキングディレクトリ

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

3.1.8 コマンドの流れ

ワークフローの該当モジュールから外部計算機のコマンドが実行されるまでの流れを下記に示す。

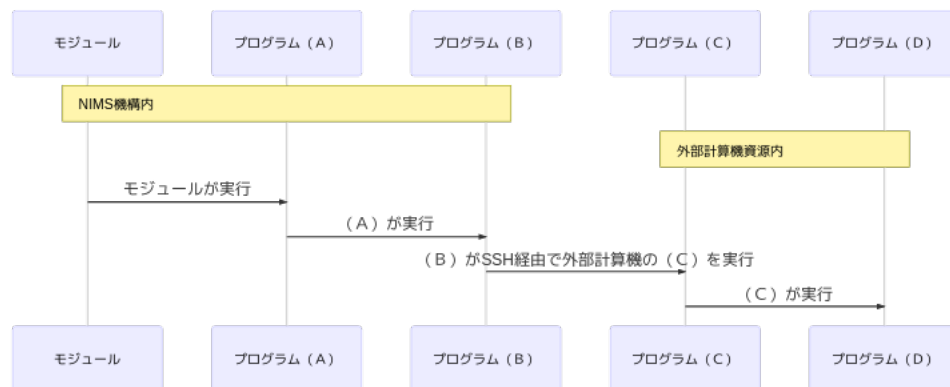


図 3.4 SSH 接続経路によるコマンド実行の流れ

- ワークフロー：予測モジュール
 - MInt システムが実行する予測モジュール
 - (A) を実行する
- プログラム (A)：kousoku_abacus_ssh_version2.sh (サンプル用)
 - MInt システムの予測モジュールが実行する。
 - 予測モジュールごとに用意する。名前は任意。使用方法 で説明する編集を行う。
 - 予測モジュール定形の処理などを行い、(B) を実行する。
 - * (B) の名前は固定である。
- プログラム (B)：execute_remote_command.sample.sh
 - (A) から実行された後、外部計算機実行のための準備を行い、SSH 経由で (C) を実行する。
 - 名前は固定である。このプログラムが外部計算機資源との通信を行う。
 - 使用方法 で説明する編集を行う。
 - * 送信するファイルはパラメータとして記述。
 - * (C) の名前は固定である。

- 受信するファイルは外部計算機資源上の計算用ディレクトリ^{*3}のファイル全部。
- プログラム (C) : `execute_remote-side_program_ssh.sh`
 - (B) から SSH で実行される。
 - 外部計算機で実行され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。
 - インストール時は `execute_remote-side_program_ssh.sample.sh`^{*4} となっている。
- プログラム (D) : `remote-side_scripts`
 - (D) から実行されるようになっており、いくつかのスクリプトを実行するよう構成されている。
 - サンプル専用であり、必ず使うものではない。(C) に依存する。

3.1.9 MInt システムと送受信されるデータ

MInt システムへ送受信されるデータは、「`execute_remote_command.sample.sh`」に記述しておく。

- 送信されるデータ
 - 「`execute_remote_command.sample.sh`」にパラメータとして記述したファイル。(モジュール内)
- 返信されるデータ
 - 計算結果としての出力ファイル。
 - * 計算専用ディレクトリを作成して計算され、そのディレクトリ以下のファイルは全て
 - * このディレクトリでの計算は、「`execute_remote-side_program_ssh.sh`」で行われるので、返信不要のファイルはあらかじめこのスクリプト終了前に削除しておくようにスクリプトを構成しておく。

※ 秘匿データを配置してあるディレクトリまたはインストール後のセットアップで実行に必要なファイル、データとして指定されたものは MInt システムで感知できないこと、およびシステム的に記録 (GPDB など) するための設定がなされていないため送り返されることは無い。

^{*3} 外部計算機では計算は `/tmp` などに一時的なディレクトリを作成し計算が実行される。

^{*4} 本システムでは、MInt システムは「`execute_remote_command.sample.sh`」を実行し、外部計算機で実行を行うプログラムとして「`execute_remote-side_program_ssh.sh`」を呼び出す。外部計算機側ではインストール後にこのファイル (インストール直後は、`execute_remote_program_ssh.sample.sh` という名前) を必要に応じて編集して使用することで、別なコマンドを記述することが可能になっている。

3.2 WebAPI 方式

続いて WebAPI 方式の説明を行う。SSH などでは直接通信が行えない組織間でも https プロトコルを利用した通信は可能なことが多く、これを利用することで外部計算機資源の有効活用できることを狙った。ただし現実的には https または TLS1.2 以上での通信しか許可されないことが多いので、本方式は https での通信のみに絞って使用することとし、そのための説明も https の使用を想定した上で行う。

3.2.1 概要

WebAPI 方式とは外部計算機資源を SSH などでは直接操作するのではなく、中間にプログラムの実行、ファイルの送受信を仲介する API を立て、MInt システム側、外部計算機資源側がその API を利用して https 通信で計算の依頼、実行などを行うシステムである。この場合、外部計算機資源側、MInt システム側（予測モジュール）は計算工程の随所で定期的に通信する必要がある（ポーリング）ので、ポーリング方式とも言う。SSH の場合と比べて外部計算機資源の利用および実行のための手続きが多くなり、用意するプログラムも複雑になる。

3.2.2 実行のイメージ

この方式では以下のようなシステムを想定している。

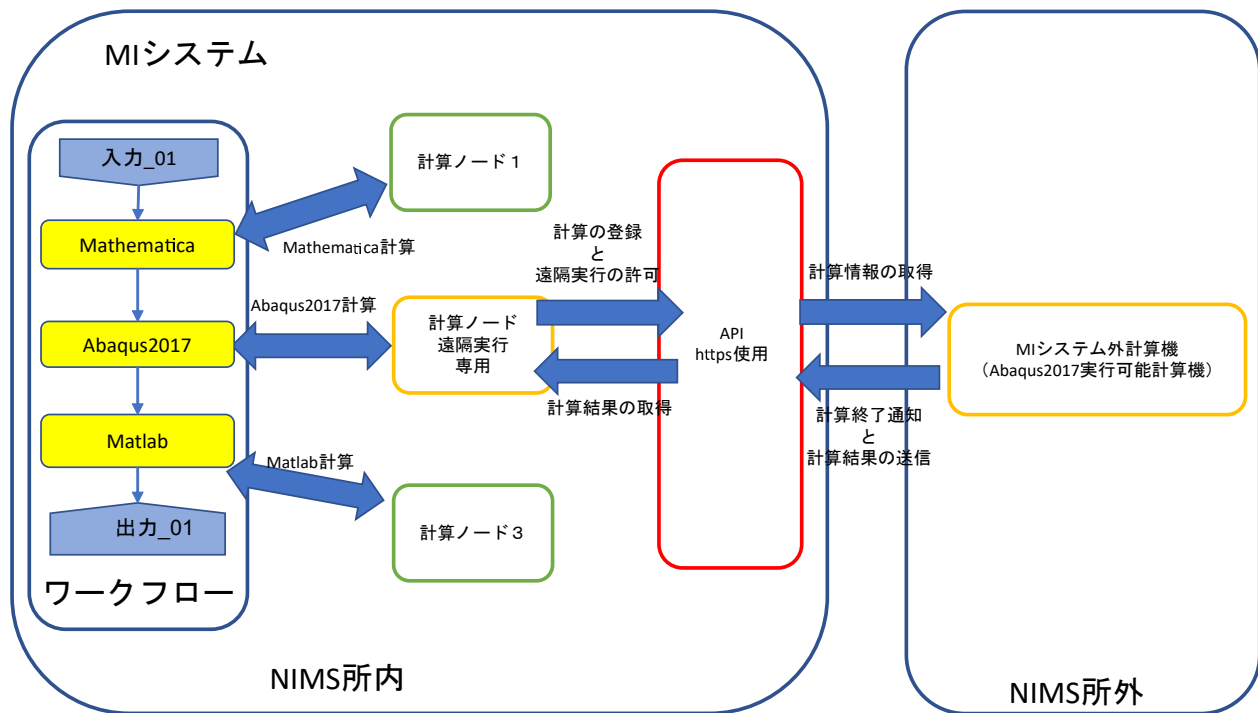


図 3.5 WebAPI 方式を利用した外部計算機資源の利用イメージ

3.2.3 WebAPI 方式の流れ

この方式での WebAPI 方式のフロー概要。

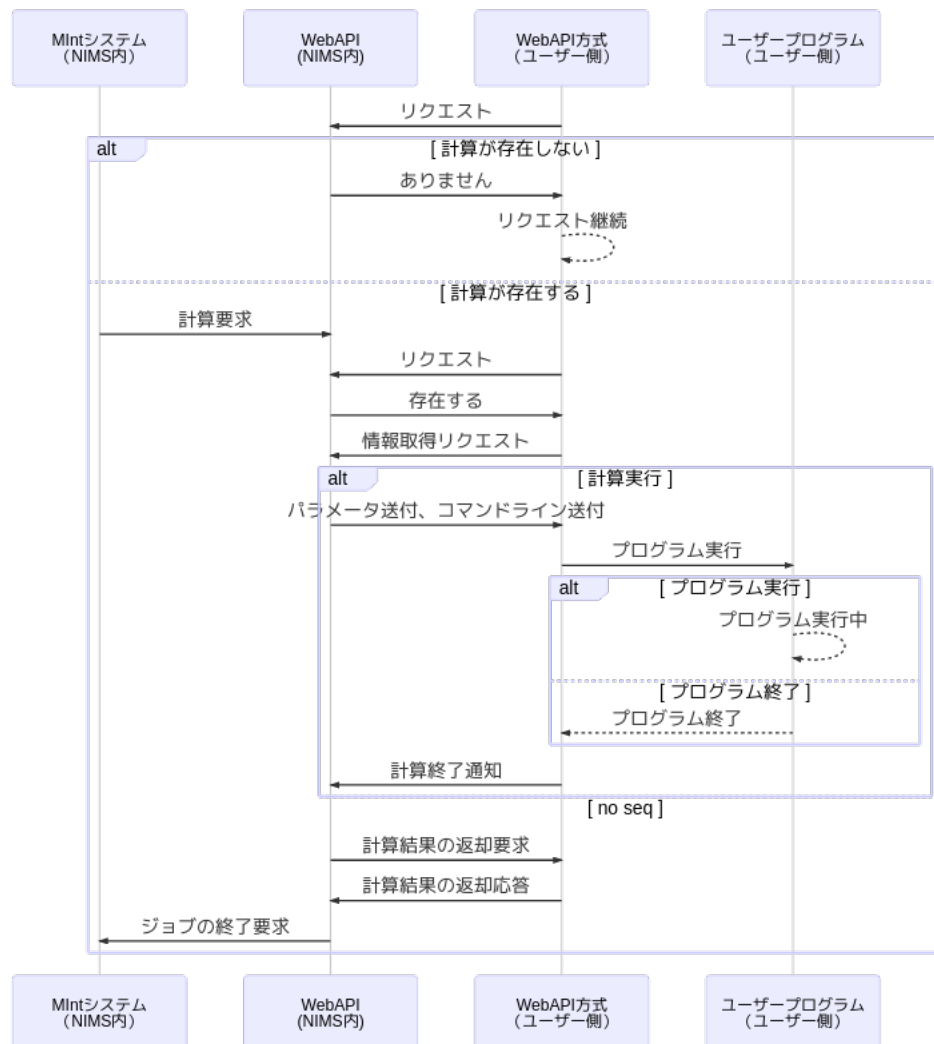


図 3.6 WebAPI 方式の流れ

3.2.4 システム要件

この方式における必要な条件を記す。おもに外部計算機資源側の条件となる。

- 双方で設定必要な事項
 - 実行可能な計算またはプログラム
 - 送受信するファイル
 - この情報を API がワークフローから遠隔計算機へ、遠隔計算機からワークフローへと受け渡す。遠隔計算機へはコマンドとパラメータ。ワークフローへは計算結果などのファイルである。
- MInt システム側
 - 外部計算機資源有効利用用の計算ノードを設置してある。(以下専用計算機または専用ノードとする)
 - 外部計算機資源有効利用モジュールを作成
 - このモジュールは専用計算機を指定して計算を行うよう実装する。
 - WebAPI 用プログラムを実行する。MInt システムへ到達可能ならどこでもよい。
 - この API プログラムはモジュールごとに専用の設定を必要とする。
 - このモジュールはこの API とだけ通信する。
- 外部計算機資源側
 - NIMS 所外にあって、https で本 API へ到達可能なネットワーク設定の場所にあること。
 - 本 API と計算を行うための WebAPI 方式用プログラムのサンプルを python で用意した。ほとんどの場合このサンプルプログラムで事足りる。
 - 用意する計算機は Linux が望ましいが、サンプルを利用する場合 python が実行可能な PC なら何でもよい。
 - 必要な資材を取得、展開。
 - 資材をローカライズ（プログラム等を環境に合わせて編集）

3.2.5 用意されているサンプルワークフロー

下記イメージの動作検証用サンプルワークフローを用意してある。

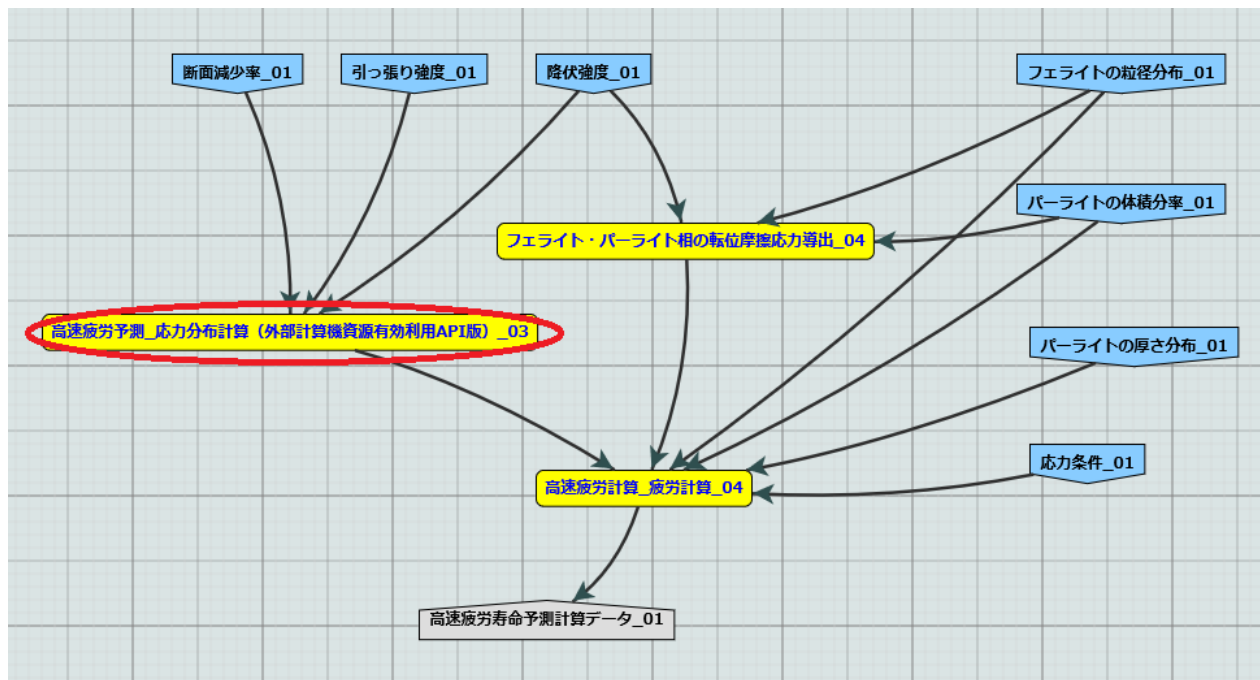


図 3.7 検証用ワークフロー

※赤枠の部分が外部計算機資源を利用するモジュールである。

3.2.6 MInt システムでのディレクトリ

MInt システム側のディレクトリ構造は以下のようになっている。

- 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。

3.2.7 外部計算機資源でのディレクトリ

外部計算機資源のディレクトリ構造は以下のようになっている。インストール方法については後述する。

- 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>
```

3.2.8 外部計算機で MInt システムから実行されるプログラム

ワークフローの該当モジュールから API 経由で外部計算機のコマンドが実行されるまでの流れを下記に示す。

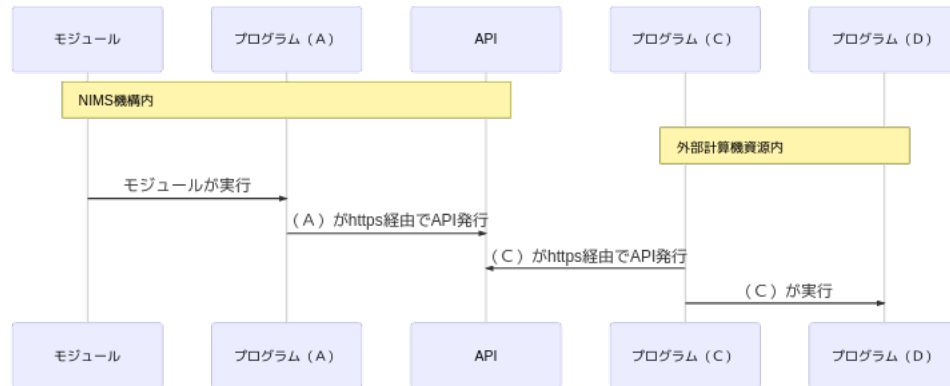


図 3.8 WebAPI 方式でのコマンドの流れ

本システムでは、MInt システムの API に設定したプログラムを外部計算機での実行に使用する。サンプルワークフローでは、「execute_remote-side_program_api.sh」となっている。外部計算機側ではインストール後にこのファイル（インストール直後は、execute_remote_program_api.sample.sh という名前）を必要に応じて編集して使用する。

3.2.9 MInt システムで送受信されるデータ

MInt システムで送受信されるデータは、MInt システム側の API と通信するモジュールの実行ファイルであらかじめ決め置く。API にはその情報によって外部計算機資源とデータのやりとりをする。この情報に必要なファイルのみ設定することで、それ以外のファイルの存在を MInt システム側で感知できず、したがって不要なファイルのやりとりは発生せず、秘匿データなどの保護が可能となる。

第 4 章

使用方法

インストールおよびプログラムの準備など説明する。SSH 方式、WebAPI 方式のそれぞれの準備から実行までを記述する。

本システムの利用者は MInt システムのアカウントは既に発行済であるものとし、その手順は記載しない。また git コマンドなどの利用方法はシステム管理者などに問い合わせることとし、ここではそれらのインストール、詳細な使用方法は言及しない。

手順は以下のようになっている。

1. 事前に決定しておく事項の列挙
2. 外部計算機側の準備
3. Int システム側の準備
4. 専用予測モジュールの準備
5. ワークフローの準備
6. WebAPI 方式の場合の準備

4.1 使用開始前に

4.1.1 事前決定事項の列挙

事前に決定しておく項目は以下の通り。

1. 方式の決定
 - 害撫計算機資源側実行ユーザーの決定または無い場合は作成。
 - MInt システム側ユーザーの決定または無い場合は作成。
 - API 方式の場合設定されている API トークンの取得。
 - 方式毎の認証情報の取り決め
2. 解析、計算の決定
 - MInt システム側で使用可能で、必要なモジュールの選定。
 - 外部計算機資源側で 1. を考慮にいれ、用意する必要のある手順の検討。
3. 実行するプログラム
 - 2. の見当の結果、外部計算機資源側で実行するプログラム及び必要なパラメータの調査。
 - MInt システムから最初に呼び出されるスクリプトの決定。
4. misc_remote_workflow リポジトリの展開場所

- クライアント側のプログラム実行場所として使用する。
 - 実行プログラム用のテンプレートなどが入っているのでこれを利用する。
5. `misrc_distributed_computing_assist_api` リポジトリの展開場所
 - WebAPI 方式の場合に必要。
 - `debug/remote-side/mi-system-reote.py` が WebAPI 方式プログラムで、これを実行しておく。

4.1.2 SSH 方式の認証情報

SSH 方式では基本的にパスワードなし接続とするため、RSA/TSA どちらかの公開鍵暗号ファイルを MInt システム側に設定する必要がある。

1. RSA 方式の公開鍵暗号ファイルの作成方法にしたがい公開鍵暗号ファイルを外部計算機資源側の実行ユーザーで作成する。
 - すべてエンターのみとする。
 - 既に存在している場合はそれを使う。
2. 作成された「`~/.ssh/id_rsa.pub`」ファイルを MInt システム担当者まで送付する。

4.1.3 WebAPI 方式の認証情報

WebAPI 方式では公開鍵ではなく、事前に取り決める「サイト名（半角英数字。文字数制限なし）」と「API アクセストークン (MInt システムで作成するユーザーアカウントに設定される)」を用意しておく。

- サイト名：他の外部系さ機資源の有効利用で使用していない名称であること。
- API アクセストークン：MInt システムに登録されている外部計算機資源側に属する人のアカウントおよびそこに設定されている API トークン。

4.1.4 外部計算機資源側の準備

1. 使用開始前に の 4. と 5. で決定した場所へリポジトリを展開する
2. 使用開始前に の 3. で決定したスクリプトを作成する。

4.1.5 MInt システム側の準備

1. 実装調査書の作成
 - 使用開始前に 2. の情報（スクリプトと送受信するパラメータ）を盛り込む。

4.1.6 専用予測モジュールの準備

1. 専用予測モジュールの作成
 - どちらの方式を採用するか。
 - 1. の情報を盛り込んだ予測モジュールを作成する。
2. ワークフローの準備
 - 2. で作成した予測モジュールを使用するワークフローを作成する。
3. SSH の場合
 - MInt 側からクライアント計算機への SSH ログインのための情報

- 鍵暗号化方式によるパスワードなし、パスフレーズなし接続が望ましい。

4. WebAPI 方式の場合

- WebAPI 方式の場合は不特定多数の利用者と API プログラムを共有するので、認証などの設定事項を MInt システム側に事前設定しておく。

4.1.7 WebAPI 方式の場合の設定事項

WebAPI 方式では、SSH とはまた違う認証情報が必要なため、それらを記述する。以下の情報は外部計算機側で WebAPI プログラムを実行する際に必要である。

- API トークン
 - 本方式では MInt システムの API 認証システムを使用しているので、そのトークンが必要となる。NIMS 側に問い合わせ取得しておく。
- ホスト情報
 - MInt システム側で API 問い合わせに対する個別の識別を行うためにサイト情報（文字列として区別できれば何でもよい）が必要である。
- MInt システムの URL
 - MInt システムの URL（エンドポイントは不要）が必要である。NIMS 側に問い合わせしておく。

4.2 SSH 方式

SSH 方式での準備を決定事項にしたがって実施する。

4.2.1 外部計算機資源側

1. misrc_remote_workflow リポジトリを以下の手順で作成しておく。

```
$ 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
abacus                                execute_remote_command.sample.sh  ↳
↳kousoku_abacus_ssh.sh
create_inputdata.py                  input_data                          ↳
↳kousoku_abacus_ssh_version2.py
execute_remote-side_program_api.sample.sh  kousoku_abacus_api_version2.py  ↳
↳kousoku_abacus_ssh_version2.sh
execute_remote-side_program_ssh.sample.sh  kousoku_abacus_api_version2.sh  ↳
↳remote-side_scripts
execute_remote_command.sample.py          kousoku_abacus_http.py
```

2. 外部計算機資源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 システム側から外部計算機資源側へ SSH ログインして最初に行われるプログラム名は「execute_remote-side_program_ssh.sh」である。このため「execute_remote-side_program_ssh.sample.sh」を「execute_remote-side_program_ssh.sh」にコピーするか、「execute_remote-side_program_ssh.sh」を独自に作成し、2. などの実行および必要な手順をスクリプト化しておく。

4.2.2 MInt システム側

1. ワークフローを作成する場合に「misrc_remote_workflow/scripts/execute_remote_command.sample.sh」を必要な名称に変更し、内容を参考にして SSH 経由実行が可能なように編集し、ワークフローから実行させる。
2. 1. を実行可能な通常どおりのワークフローを作成する。作成方法に差は無い。

4.3 WebAPI 方式

4.3.1 外部計算機資源側

1. misrc_distributed_computing_assist_api リポジトリを以下の手順で作成しておく。

```
$ git clone https://gitlab.mintsys.jp/midev/misrc_distributed_computing_assist_api
$ cd misrc_distributed_computing_assist_api
$ ls
```

(次のページに続く)

(前のページからの続き)

```
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. my-system-remote.py を実行しておく。

```
$ python mi-system-remote.py rme-u-tokyo https://nims.mintsys.jp <API token>
```

4.3.2 MInt システム側

1. misrc_distributed_computing_assist_api リポジトリを展開。
2. mi_dicomapi.py が本体であるが、まだ動作させてなければ、mi_distributed_computing_assist.ini に外部計算機資源側の設定を実施する。動作させていたら、設定の再読み込みを実施する。

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

3. まだ動作していなかったら、動作させて待ち受け状態にしておく。

```
$ python mi_dicomapi.py
```

4.4 ワークフローについて

外部計算機資源利用を行うワークフローの作成の仕方を記述する。

4.4.1 共通事項

SSH 方式と WebAPI 方式の両方に共通する事項である。

- 予測モジュール
 - 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。
 - 外部計算機資源側で別途 Torque などのバッチジョブシステムに依存する。

4.4.2 SSH 方式

予測モジュールの実行プログラムから `misrc_remote_workflow/scripts/execute_remote_command.sample.sh` またはこのファイルを専用に別名コピー編集したものを必要なパラメータとともに実行するように構成する。

4.4.3 WebAPI 方式

予測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 サンプル

`misrc_remote_workflow` リポジトリにある、`sample_data` ディレクトリにテストで使用したワークフロー実行用のサンプルファイルが用意されている。これを利用してワークフローおよび外部計算機側の動作の実行テストが可能である。

また、`misrc_remote_workflow/scripts` にこの時の予測モジュール実行プログラムがある。これを参考に別な予測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。

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

以上

参考文献

[activities_of_NIMS] NIMS の取り組みについて.pdf