国立研究開発法人　理化学研究所　御中

2020年度InSitu処理向け三次元可視化  
フレームワークのプロトタイプ整備

作業報告書

第1.0版

富士通株式会社

2021年1月

改版履歴

|  |  |  |
| --- | --- | --- |
| リリース | 版数 | 備考 |
| 2021/01/27 | 1.0 | 初版 |

# 目次

[目次 3](#__RefHeading___Toc6684_1389316416)

[1. はじめに 4](#__RefHeading___Toc6686_1389316416)

[2. システム概要 5](#__RefHeading___Toc6688_1389316416)

[2.1. ソフトウエア構成 5](#__RefHeading___Toc6690_1389316416)

[2.2. 動作環境とインストール作業 5](#__RefHeading___Toc6692_1389316416)

[2.2.1. 必要な動作環境 5](#__RefHeading___Toc6694_1389316416)

[2.2.2. インストール 5](#__RefHeading___Toc6696_1389316416)

[3. 操作説明 7](#__RefHeading___Toc6698_1389316416)

[3.1. 概要 7](#__RefHeading___Toc6700_1389316416)

[3.2. Temporal Buffer 7](#__RefHeading___Toc883_3236865239)

[3.2.1. 実行方法 7](#__RefHeading___Toc6702_1389316416)

[3.2.2. コマンドラインオプション 7](#__RefHeading___Toc6704_1389316416)

[3.2.3. 終了方法 8](#__RefHeading___Toc6704_13893164162)

[3.3. TB2C server 8](#__RefHeading___Toc6728_1389316416)

[3.3.1. 実行方法 8](#__RefHeading___Toc6730_1389316416)

[3.3.2. コマンドラインオプション 8](#__RefHeading___Toc6704_13893164161)

[3.3.3. 終了方法 8](#__RefHeading___Toc6704_138931641621)

[3.4. TB2C client 8](#__RefHeading___Toc6728_13893164161)

[3.4.1. 実行方法 8](#__RefHeading___Toc6730_13893164161)

[3.4.2. コマンドラインオプション 9](#__RefHeading___Toc6704_138931641611)

[3.4.3. 操作方法 9](#__RefHeading___Toc6704_1389316416111)

[3.4.4. 終了方法 11](#__RefHeading___Toc6704_1389316416211)

# はじめに

本書は、国立研究開発法人理化学研究所向け「2020年度InSitu処理向け三次元可視化フレームワークのプロトタイプ整備」の作業報告書です。

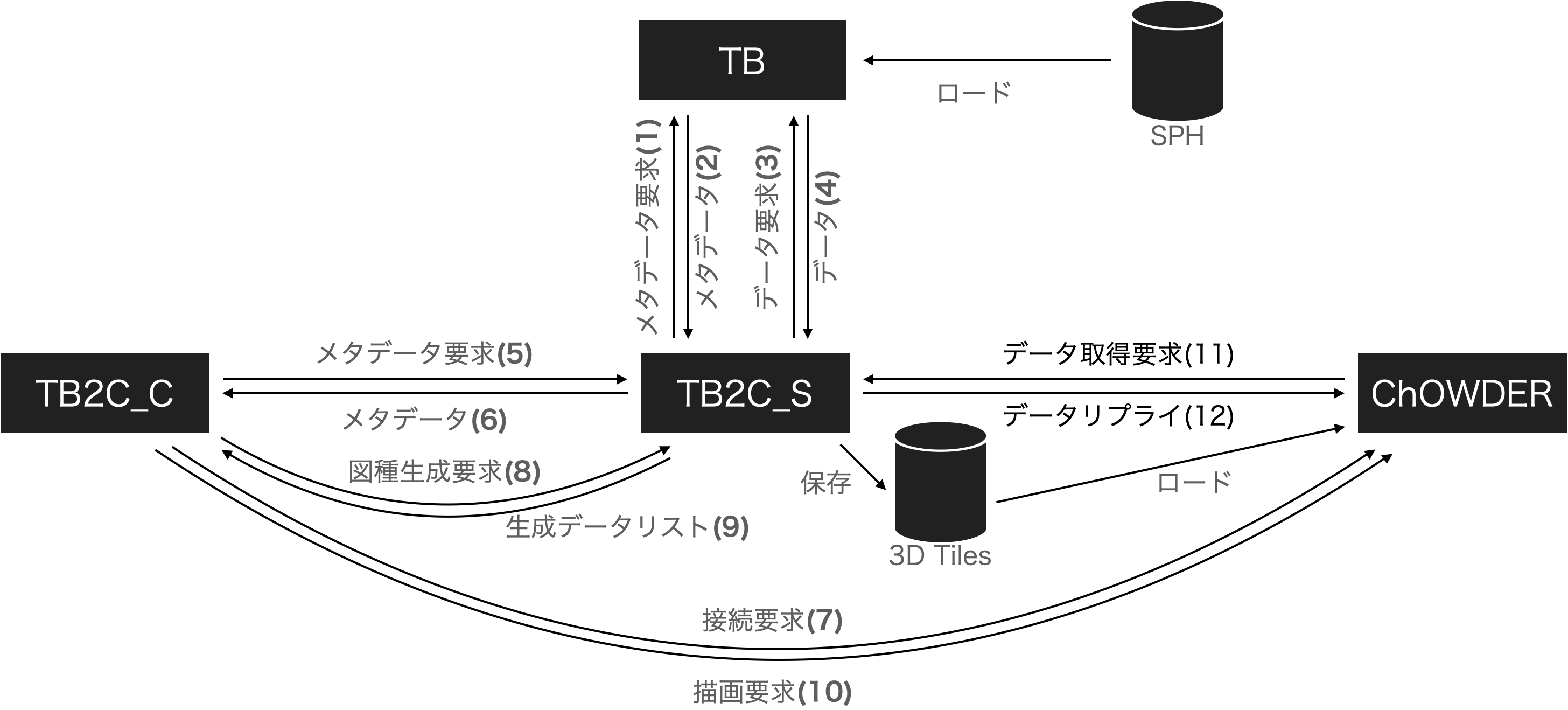
本作業で実装したTB2Cの実装報告および動作検証報告を記述しています。

# **実装報告**

# システム設計

# **システムの構成**

TB2Cは、Temporal Buffer (TB)とTB2C\_server (TB2C\_S)およびTB2C client (TB2C\_C)から構成されるシステムであり、ChOWDERに接続して動作します(下図参照)。



Temporal Buffer(TB)は、SPHフォーマットで用意された時系列の数値シミュレーション結果データを読み込んでバッファリングし、外部からの要求に応じて必要な時刻スライス・物理量のデータを提供します。

TB2C client(TB2C\_C)はユーザーが直接操作するGUIプログラムであり、可視化パラメータの設定や時刻スライスの指定、視点の変更等の操作を行い、ChOWDERに対する表示更新要求を行います。

TB2C server(TB2C\_S)は、TB2C\_Cからの要求に応じて、TBが保持するデータを取得して可視化図種の生成を行い、ChOWDER表示用に3D-Tiles形式のファイルに出力します。

TBおよびTB2C serverは、ChOWDERと同一のマシン上で動作することを前提としています。TB2C clientは別のマシン上で動作し、ChOWDERおよびTB2C serverと通信を行います。

# **プログラム間の通信データ**

以下に、各プログラム間で通信されるデータを示します。

(1) TB2C\_serverからTBへのメタデータ要求

 TBのポートへのHTTP GET

path="/",

param:なし

(2) TBからTB2C\_serverへ返されるメタデータ

 JSON[{"id":データID,

　　　"uri":先頭SPHファイルのURI,

　　　"type":"SPH",

　　　"dims":[I,J,K],

　　　"datalen":ベクトル長,

　　　"bbox":[[x0,y0,z0],[x1,y1,z1]],

 　　　"steps":ステップ数,

　　　"timerange":[開始時刻,終了時刻],

　　　"vrange":[最小値,最大値] }, …]

TBが保持するデータがリスト化されたJSONが返される(今回の実装では１個のみ)

(3) TB2C\_serverからTBへのデータ要求(指定したタイムステップのデータを取得する)

 TBのポートへのHTTP GET

path="/data",

param:id=データID,

step=取得するステップ番号

(4) TBからTB2C\_serverへ返されるデータ

 JSON{"type":"SPH",

"step":ステップ番号,

"data":{データ本体} }

{データ本体}は、SPHクラスを base64でエンコードしてJSON化したもの

(5) TB2C\_clientからTB2C\_serverへのメタデータ要求

 TB2C\_serverのポートへのHTTP GET

path="/",

param:なし

(6) TB2C\_serverからTB2C\_clientへ返されるメタデータ

 JSON{"id":データID,

　　　"uri":先頭SPHファイルのURI,

　　　"type":"SPH",

　　　"dims":[I,J,K],

　　　"datalen":ベクトル長,

　　　"bbox":[[x0,y0,z0],[x1,y1,z1]],

 　　　"steps":ステップ数,

　　　"timerange":[開始時刻,終了時刻],

　　　"vrange":[最小値,最大値] ,

　　　"vistype":[“isosurf”]}

vistypeには、TB2C\_serverがサポートする可視化図種のリストが入る(今回の実装では“isosurf”のみ)

(7) TB2C\_clientからChOWDERへの接続要求

 ChOWDERへのWebSocket通信

path="ws://chowder\_host/v2",

method:"AddContent"

今回の実装では、chowder\_hostは“localhost”のみ

(8) TB2C\_clientからTB2C\_serverへの図種生成要求

 TB2C ServerのポートへのHTTP PUT

path="/visualize"

param:step=対象ステップ番号,

　　　 vistype="isosurf",

　　　 visparam={"value":等値面の値}

 SPHがベクトルデータの場合は、データの L2ノルムで等値面を生成する

TB2C Serverによる図種生成完了を待ってChOWDERへの描画要求(9)を行う

(9) TB2C\_serverからTB2C\_clientへ返される生成データリスト

JSON[{tiledLayer}, …]

TB2C\_serverで生成された3D-Tilesデータの個数分のtiledLayerのリストが返される

(10) TB2C\_clientからChOWDERへの描画要求

ChOWDERへのWebSocket通信

path="ws://chowder\_host/v2",

method:"UpdateMetaData"

タイムステップまたは等値面の値が変更された場合は、アップデートIDを変更する

視界が変更された場合は、”cameraWorldMatrix”を更新する

(11) ChOWDERからTB2C\_serverへのデータ取得要求

TB2C\_serverのポートへのHTTP GET

path="/visualized/...",

param:なし

(12) TB2C\_serverからChOWDERへのデータリプライ

要求されたファイル

今回の実装ではファイル渡し

# **モジュール設計**

# **Temporal Buffer**

# **TB2C server**

# **TB2C client**

# **動作検証****報告**

# **概要**

「富岳」を想定したHPCI計算資源を用いての動作検証として、九州大学ITOシステムの占有フロントエンドノード(ITOフロントエンド)にTB2CシステムおよびChOWDERシステムをインストールし、ここで動作するサーバー群とローカルPC上で動作するTB2Cクライアントを接続して動作検証を行います。

この際に、ChOWDERディスプレイとして、理化学研究所R-CCSのタイルドディスプレイ装置に表示するWebブラウザを使用し、ChOWDERのレンダリング時間計測機能を使用してレンダリング性能を測定します。

なお、ITOフロントエンドに割り当てられるIPアドレスはプライベートアドレスであり、ITOログインノードを介したssh接続しか行うことができないため、動作検証作業に際してはITOログインノードへの通信をsshのポートフォワーディング機能を使用してITOフロントエンドに転送する設定を行います。

また、ITOフロントエンドではroot権限が与えられないため、TB2CシステムおよびChOWDERシステムのインストールに関わる全てのソフトウエアのインストールはユーザー環境に行います。

ChOWDERが使用する通信ポートは、デフォルトでは80番(および443番)ですが、今回の作業ではroot権限を必要としない1024番以上のポート(8080番)を使用します。

# **ITO****フロントエンドでの環境構築**

# **ノード構成**

* ノードテンプレート：BLGI(ベアメタル)
* OS：Red Hat Enterprise Linux Server release 7.3
* CPU：36コア、主記憶：384GiB

# **Spackのインストールおよび設定**

ITOフロントエンドでPython3、Node.jsおよびこれらの関連モジュールをユーザー環境にインストールするため、パッケージ管理システムSpackを使用します。

以下のコマンドをITOフロントエンド上で実行することで、Spackがインストールされます。

　cd $HOME

　git clone https://github.com/spack/spack.git

これにより、ホームディレクトリ配下にspackというディレクトリが作成され、ここにSpackの実行環境がダウンロードされます。ここで、以下のコマンドを実行するとspackコマンドが利用可能になります。

　source $HOME/spack/share/spack/setup-env.sh

ITOフロントエンドでは、environment-moduleシステムで複数のコンパイラが使用可能となっていますが、今回の作業ではgcc-9.2.0を使用します。

以下のコマンドをITOフロントエンド上で実行することで、Spackがコンパイラとしてgcc-9.2.0を使用可能になります。

　module load gcc/9.2.0

　spack compiler find

上記のコマンドを実行することにより、$HOME/.spack/linux/compiler.yamlファイルが作成され、Spackのコンパイラーリストにgcc@9.2.0が追加されます。

最後に、$HOME/.spack/packages.yamlファイルを作成し、以下の内容を記述します。

packages:

perl:

externals:

- spec: "perl@5.32.0"

prefix: /usr

buildable: False

all:

compiler: ['gcc@9.2.0', 'gcc@4.8.5', 'gcc@4.4.7']

target: [x86\_64]

これにより、Spackは/usrにインストールされているPerlを使用し、パッケージとしてインストールすることはなくなります。

# **Node.jsおよびnpmのインストール**

Spackを使用してNode.jsおよびnpmのインストールを行うには、以下のコマンドを実行します。

　spack install node-js%gcc@9.2.0

　spack install npm%gcc@9.2.0

インストール終了後は、以下のコマンドを実行するとnodeおよびnpmコマンドが利用可能になります。

　spack load node-js%gcc@9.2.0

　spack load npm%gcc@9.2.0

次に、インストールしたnpmを使用してobj23dtilesをインストールします。以下のコマンドを実行します。

　npm install -g obj23dtiles

# **Python3およびPythonモジュール群のインストール**

Spackを使用してPython3のインストールを行うには、以下のコマンドを実行します。

　spack install python%gcc@9.2.0

インストール終了後は、以下のコマンドを実行するとpython3およびpip3コマンドが利用可能になります。

　spack load python%gcc@9.2.0

次に、インストールしたPython3(pip3コマンド)を使用して、TB2Cの実行に必要なPythonモジュール群をインストールします。以下のコマンドを実行します。

　pip3 install scikit-image

　pip3 install wxPython

　pip3 install pyOpenGL

　pip3 install websocket-client

# **Spack環境のbash設定**

Spackを使用してインストールした環境を、次回ログイン時にも再現できるように、$HOME/.bash\_profileに以下の記述を追加しておきます。

if [ -d $HOME/spack ]; then

source $HOME/spack/share/spack/setup-env.sh

spack load python%gcc@9.2.0

spack load node-js%gcc@9.2.0

spack load npm%gcc@9.2.0

fi

# **ChOWDERのインストール**

ChOWDERのインストールは、ChOWDERのgithubリポジトリよりソースをダウンロード(clone)し、インストールスクリプトを実行することで行います。

まず、ソースのダウンロードを行います。以下のコマンドを実行します。

　cd $HOME

　git clone -b 202009 http://github.com/digirea/ChOWDER.git

これにより、ホームディレクトリ配下のChOWDERディレクトリに、ChOWDERのソース一式(202009ブランチ)がダウンロードされます。

次に、インストールスクリプトを実行します。以下のコマンドを実行します。

　cd $HOME/ChOWDER/bin

　sh ./install.sh

通常、インストールスクリプトの実行でChOWDERのインストールは完了しますが、今回使用しているChOWDERのブランチ(202009ブランチ)では、CentOS 7用のRedisサーバーがインストールされません。そこで、以下のようにRedisのソースコードをダウンロードし、Redisサーバーのインストールを行います。

wget http://download.redis.io/releases/redis-5.0.5.tar.gz

　tar xvfz redis-5.0.5.tar.gz

　cd redis-5.0.5

　make

コンパイルが終了したら、Redisサーバーの実行ファイルをChOWDER環境下にコピーします。

　cp src/redis-server $HOME/ChOWDER/redis/

# **TB2Cのインストール**

TB2Cのインストールは、TB2Cの提供ファイル(TB2C-1.x.tar.gz)をsftpでITOフロントエンドにコピーし、任意のディレクトリで展開することで行えます。

ここでは、ホームディレクトリ配下のTB2Cディレクトリ以下に展開します。

　cd $HOME

　tar xvfz TB2C-1.x.tar.gz

# **実施方法**

* + 1. **ITOフロントエンドでのサーバー実行**

ITOフロントエンド上で、Tenporal Buffer、TB2C serverおよびChOWDERを動作させておく必要があります。

尚、以下の説明ではTB2C展開ディレクトリは$HOME/TB2C、ChOWDER展開ディレクトリは$HOME/ChOWDERとしています。

(1) Temporal Buffer

ITOフロントエンド上のTB2C展開ディレクトリに移動し、以下のコマンドを実行します。

　cd $HOME/TB2C

　python3 python/TB.py -j data/concat\_input\_p.json

(2) TB2C server

ITOフロントエンド上のTB2C展開ディレクトリに移動し、以下のコマンドを実行します。

　cd $HOME/TB2C

　python3 python/TB2C\_server.py --odir $HOME/ChOWDER/public/data ¥

　　　--dx 2 --dy 2

(3) ChOWDER

ITOフロントエンド上のChOWDER展開ディレクトリ/bin に移動し、以下のコマンドを実行します。

　cd $HOME/ChOWDER/bin

　sh ./run.sh

# **ITOログインノードでのポートフォワーディング設定**

ITOログインノード(ito.cc.kyushu-u.ac.jp)上では、外部からの8080番ポートへの接続をITOフロントエンドの8080番ポートに、また外部からの4000番ポートへの接続をITOフロントエンドの4000番ポートに、それぞれポートフォワーディングする必要があります。

これには、ITOログインノード上で以下のsshコマンドを実行します。ここで、xx.xx.xx.xx にはITOフロントエンドノードのIPアドレスを指定します。

■ 8080番ポートのポートフォワーディング

　ssh -g -L 8080:xx.xx.xx.xx:8080 xx.xx.xx.xx

■ 4000番ポートのポートフォワーディング

　ssh -g -L 4000:xx.xx.xx.xx:4000 xx.xx.xx.xx

これらは、ITOログインノードに別個にログインした別のターミナルで実行する必要があります。

また、ITOログインノードはito-1とito-2の２台のマシンが負荷分散のために切り替えられて使用されているため(本書執筆時点)、ito-1とito-2の両方のマシン上で上記のコマンドを実行する必要があります。

# **ローカルPCでのポートフォワーディング設定**

ローカルPC(TB2C clientを実行するマシン)上では、80番ポートへの接続をITOログインノードの8080番ポートに、また4000番ポートへの接続をITOグインノードの4000番ポートに、それぞれポートフォワーディングする必要があります。

これには、ローカルPC上で以下のsshコマンドを実行します。

■ 80番ポートのポートフォワーディング

　ssh -i 秘密鍵 -L 80:ito.cc.kyushu-u.ac.jp:8080 ユーザー名@ito.cc.kyushu-u.ac.jp

■ 4000番ポートのポートフォワーディング

　ssh -i 秘密鍵 -L 4000:ito.cc.kyushu-u.ac.jp:4000 ユーザー名@ito.cc.kyushu-u.ac.jp

これらは、ローカルPC上の別個のターミナルで実行する必要があります。

# **ChOWDERコントローラーの実行**

ローカルPC上でWebブラウザを起動し、以下のURLに接続することでChOWDERコントローラーが表示されます。

　http://localhost:80/

80番ポートのポートフォワーディングがされているので、ITOフロントエンド上のChOWDERサーバーに接続しています。

# **TB2C clientの実行**

ローカルPC上でTB2C clientを実行し、ITOフロントエンド上のTB2C serverおよびChOWDERサーバーに接続させます。ローカルPC上のTB2C展開ディレクトリに移動し、以下のコマンドを実行します。

　python3 python/TB2C\_client.py -s http://localhost:4000/ -c localhost

上記コマンドを実行すると、TB2C clientのウインドウが表示され、ChOWDERのAPIUserのパスワード入力が求められます。パスワードを入力すると、ChOWDERコントローラーにはContent IDがtb2c\_3dtileであるコンテンツが登録されます。

ここで、TB2C clientのGUIでtimestep indexを40に、isosurf valueを0.0に設定すると、下図のような表示になります。



# **ChOWDERディスプレイのレンダリング時間測定**

ChOWDERディスプレイの接続を行い、ローカルPC上のTB2C展開ディレクトリに移動して、以下のコマンドを実行すると、ChOWDERディスプレイのレンダリング時間の測定が行われます。

　python3 python/chowder\_measure.py