TSUBAME4.0利用の手引き

TSUBAME Computing Services, Center for Information Infrastructure

2025-04-24

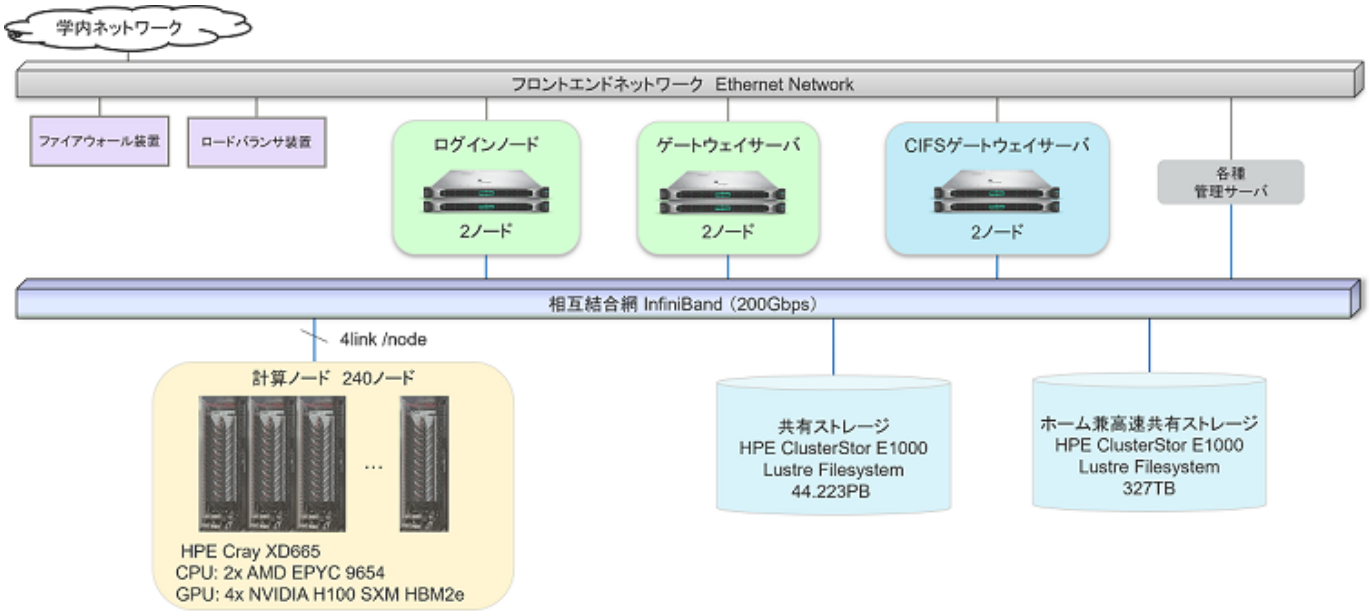
Table of contents

1. TSUBAME4.0概要

1.1. システム概念

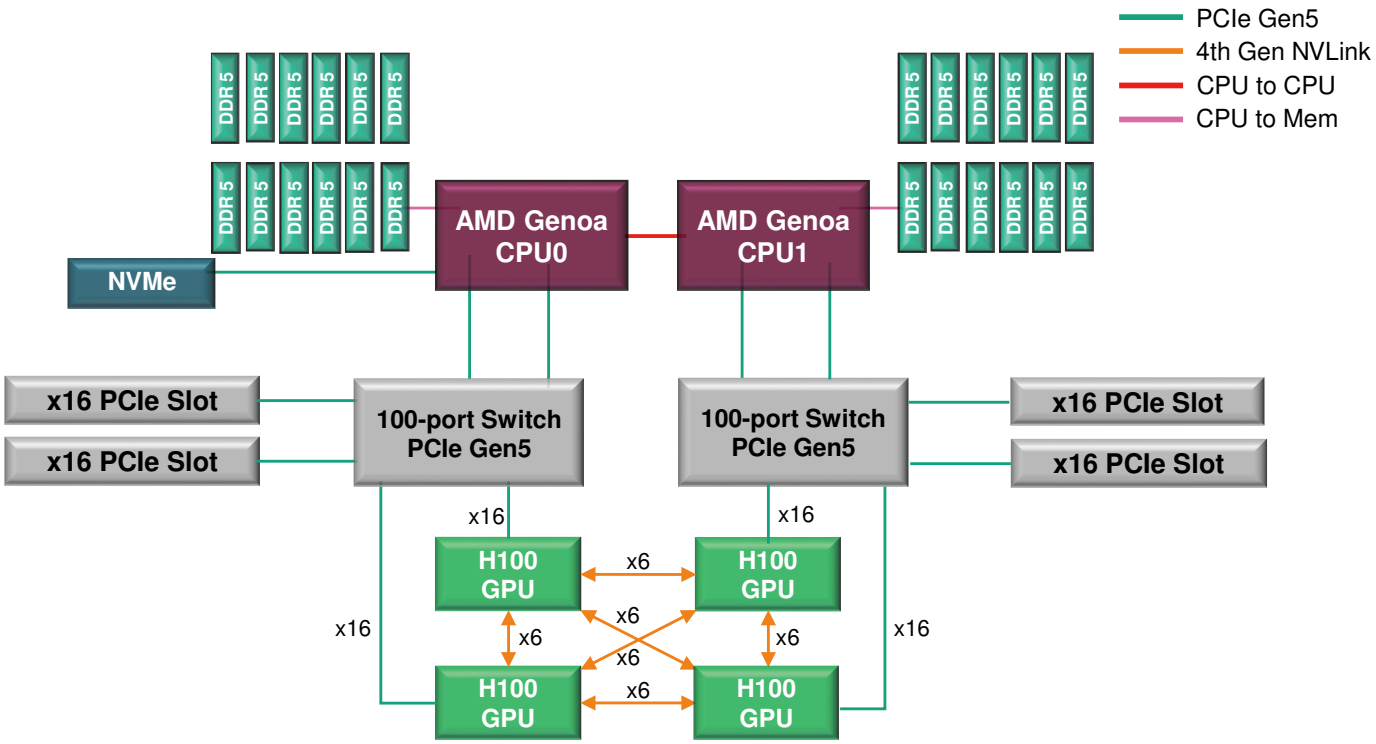
本システムは、東京科学大学における種々の研究・開発部門から利用可能な共有計算機です。 本システムの倍精度総理論演算性能は66.8PFLOPS、半精度総理論演算性能は952PFLOPS、総主記憶容量は180TiB、総SSD容量は327TB、総ハードディスク容量は44.2PBです。 各計算ノード及びストレージシステムは InfiniBand による高速ネットワークに接続され、SINET6を経由し100Gbpsの速度で東京科学大すずかけ台キャンパスから直接インターネットに接続されます。

TSUBAME4.0の全体概念を以下に示します。



1.2. 計算ノードの構成

本システムの計算ノードはHPE Cray XD665 240ノードで構成された大規模クラスタシステムです。 1台の計算ノードには、AMD EPYC 9654 (96コア, 2.4GHz) を2基搭載し、総コア数は46,080コアとなります。 また、主記憶容量は計算ノードあたり768GiBを搭載し、総主記憶容量は、180TiBとなります。 各計算ノードは、InfiniBand NDR200 インタフェースを4ポート有しており、InfiniBand スイッチによりファットツリーで接続されます。



TSUBAME4.0は240台の HPE Cray XD665 計算ノードからなり、各計算ノードの基本スペックは次の通りです。

構成要素	製品・性能
CPU	AMD EPYC 9654 2.4GHz × 2 Socket
コア数/スレッド数	96コア / 192スレッド × 2 Socket
メモリ	768GiB (DDR5-4800)
GPU	NVIDIA H100 SXM5 94GB HBM2e × 4
SSD	1.92TB NVMe U.2 SSD
インターコネクト	InfiniBand NDR200 200Gbps × 4

1.3. ソフトウェア構成

本システムのオペレーティングシステム(OS)は、下記の環境を有しています。

・RedHat Enterprise Linux Server 9.3

OS構成は、サービス実行形態に応じて動的に変更されます。

また、本システムで利用可能なアプリケーションソフトウェアに関しては、 商用アプリケーション、フリーウェアを参照ください。

1.4. ストレージ構成

本システムでは、様々なシミュレーション結果を保存するための高速・大容量のストレージを備えています。 計算ノードでは、高速ストレージ領域、大容量ストレージ領域、Homeディレクトリ、および共通アプリケーション配備は、Lustre ファイルシステムによりファイル共有されています。 また、各計算ノードにローカルスクラッチ領域として1.92TBのNVMe SSDが搭載されています。(一部領域は別の用途で利用しています)

本システムで利用可能な、各ファイルシステムの一覧を以下に示します。

用途	マウント	容量	ファイルシステム
高速ストレージ領域 Homeディレクトリ	/gs/fs /home	372TB	Lustre
大容量ストレージ領域 共有アプリケーション配備	/gs/bs /apps	44.2PB	Lustre
ローカルスクラッチ領域	/local	各ノード1.62TiB	xfs　SSD)

ローカルスクラッチ領域の容量については、利用可能な資源タイプ をご覧下さい。

2. 利用開始にあたって

Info

本ページのコマンドライン例では、以下の表記を使用します。
[login]$: ログインノード
[rNnN]$: 計算ノード
[login/rNnN]$: ログインノードまたは計算ノード
[yourPC]$: ログインノードへの接続元環境

2.1. アカウントの取得

本システムを利用するには、予め利用申請を行い、ユーザIDを取得する必要があります。

利用者区分に応じて必要な操作・手続きが異なりますので、詳細はアカウント取得方法をご参照ください。

2.2. ログイン方法

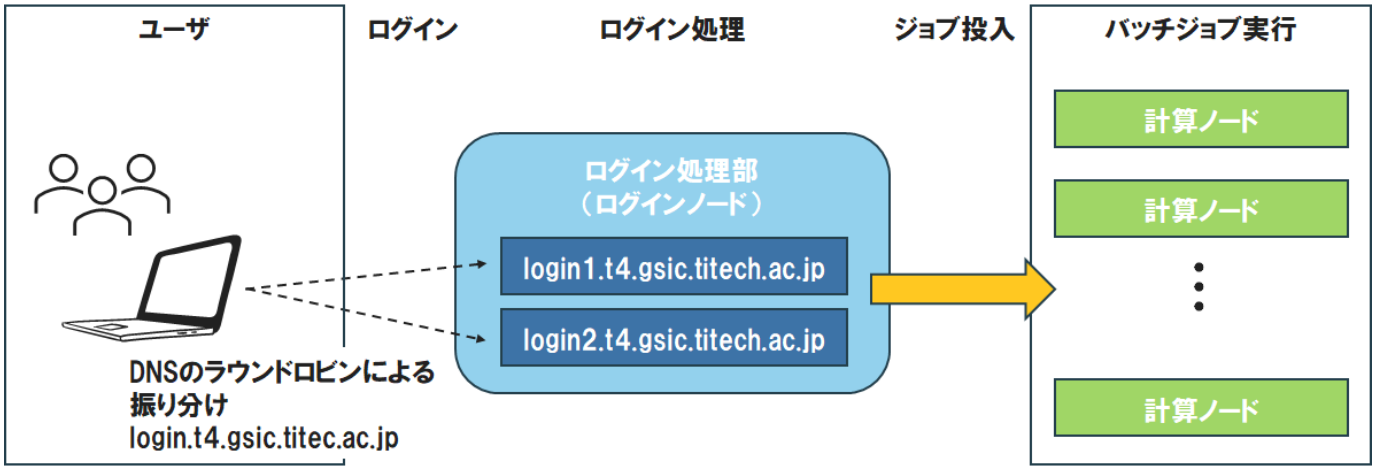
ログインノードにアクセスするためには、ログインに使うSSH公開鍵をアップロードする必要があります。 公開鍵の登録の操作は、TSUBAME4.0 ポータル利用の手引き SSH公開鍵の登録を参照ください。

本システムを利用するには、まずログインノードにログインする必要があります。 ログインノードへのログインは、DNSのラウンドロビンによる自動振り分けが行われます。

Warning

ログインノードは複数のユーザで共用されているため、負荷のかかる操作は行わないでください。

利用イメージを以下に示します。



ログイン先には、SSHで接続します。また、ファイル転送はSFTPで接続します。

```
login.t4.gsic.titech.ac.jp
```

任意のログインノードにログインしたい場合は、以下のホスト名(FQDN)を指定してください。

```
login1.t4.gsic.titech.ac.jp
login2.t4.gsic.titech.ac.jp
```

Linux/Mac/Windows(Cygwin)からX転送オプションを有効にして接続するには以下の例のようになります。

例)TSUBAMEログイン名が `ux00000`、秘密鍵が `~/.ssh/t4-key` の場合

```
[yourPC]$ ssh ux00000@login.t4.gsic.titech.ac.jp -i ~/.ssh/t4-key -YC
```

鍵ペアの格納場所を標準のパス・ファイル名とした場合には `-i` オプションは指定不要です。

最初にログインする際、クライアントの設定によっては以下のようなメッセージが出ることが有ります。 その場合は、下記出力例と fingerprint が一致することを確認のうえ、 `yes` と入力してください。
(fingerprintの種類は、お手元の ssh コマンドのバージョンで異なります)

```
The authenticity of host 'login.t4.gsic.titech.ac.jp (131.112.133.51)' can't be established.
ECDSA key fingerprint is SHA256:CyUFJvJ9ExAE4QEM3BTcKHZ2g7Mw57NnCzbvqIM0ewo.
ECDSA key fingerprint is MD5:d0:99:50:c7:ae:d4:32:62:a6:b8:5e:c0:bc:11:37:96.
ED25519 key fingerprint is SHA256:l++Qi9lvnHIRQbQ+638MgWsZdSyZvdAWicwDsYq5KBI.
Are you sure you want to continue connecting (yes/no)?
```

2.2.1. ログインノードにおける高負荷プログラムの実行制限について

ログインノード(login, login1, login2)は多数のユーザが同時に利用しているため、CPUを占有するプログラムを実行しないでください。 並列計算、長時間な計算は計算ノードを利用してください(qsub/qrshコマンドを利用する)。 以下に判断の目安を例示します。 ここで許可されている、もしくは触れていない項目についても、他ユーザの利用の妨げとなっているプログラムについては、システム管理者の判断で予告なく停止・削除させていただきます。

基本的には問題ないこと

・ファイルの転送・展開 (scp, sftp, rsync, tarなど)

・プログラムのコンパイル(ただし並列コンパイルなど多数の資源を一度に使う場合は計算ノードをご利用ください)

行わないで欲しいこと

・商用アプリケーション、フリーソフトウェアおよび自作プログラムによる計算の実行

・10分を超えるプログラムの実行(ファイル転送を除く)

・並列処理を行うプログラム(pythonによるものやMPIを含む)の実行

・メモリを大量に消費するプログラムの実行

・多数のプロセスの同時実行 (並列コンパイルなど)

・常駐するプログラムや、停止時に方法の如何に関わらず自動で再実行されるプログラム(VSCode ServerやJupyter Notebookなど)

・その他、CPUに長時間負荷がかかる作業

Jupyter Lab, code-server(VS codeのクローン)は、Open OnDemand経由で計算ノードで実行することができます。

ログインノードでは、1プロセスあたり4GBのメモリ制限を課しています。 また、システムに負荷を与えているプログラムはシステム管理者によって予告なく停止させていただきますのでご注意ください。

ログインノードが高負荷で作業しづらい時や、負荷のかかる作業を行うときは、ジョブスケジューラ経由でインタラクティブジョブとして実行してください。

2.3. パスワード管理

本システムのユーザアカウントはLDAPサーバで管理され、システム内の認証はSSHの鍵認証で行っています。 このため、計算ノードの利用にあたってパスワードを意識する必要はありませんが、 学内から高速ストレージへのアクセスなどパスワードが必要になるケースがあります。

パスワードの変更が必要になる場合は、 TSUBAME4.0利用ポータルから行ってください。 パスワードのルールについては、TSUBAME4.0利用ポータルのパスワード設定のページをご覧ください。

2.4. ログインシェルの変更

ユーザ登録の時点で各ユーザアカウントのログインシェルはbashとなっています。 デフォルトのログインシェルを変更するにはchshコマンドを利用ください。 利用可能なログインシェルはbash, tcsh, zshとなります。 引数なしのchshコマンドで利用可能なログインシェルを確認することができます。

```
[login]$ chsh
Usage: chsh shell(/bin/bash /usr/bin/zsh /usr/bin/tcsh).
```

変更が反映されるまで最大1時間程度かかる場合があります。

以下は、 ログインシェルをtcshに変更する例です。

```
[login]$ chsh /usr/bin/tcsh
Please input Web Portal Password(not SSH Passphrase)
Enter LDAP Password: xxxxxx        ← パスワードを入力してください

Changing shell succeded!!
```

2.5. TSUBAMEポイントの確認

コマンドでのTSUBAMEポイントの確認は `t4-user-info group point` コマンドにて確認できます。 以下は、TESTGROUPのTSUBAMEポイントを確認する例です。

```
[login]$ t4-user-info  group point -g  TESTGROUP
gid     group_name                      deposit      balance
-----------------------------------------------------------
xxxx  TESTGROUP                            10          5000
```

参加しているTESTGROUPの仮ポイントが10ポイント、残TSUBAMEポイントが5000ポイントである状況が確認できます。

3. ストレージ環境

本ページのコマンドライン例では、以下の表記を使用します。
[login]$: ログインノード
[rNnN]$: 計算ノード
[login/rNnN]$: ログインノードまたは計算ノード
[yourPC]$: ログインノードへの接続元環境

本システムでは、Homeディレクトリ以外にも 高速および大容量ストレージ領域のLustreファイルシステム、ローカルスクラッチ領域のSSD領域など様々な並列ファイルシステムを利用することができます。

3.1. Homeディレクトリ

HOMEディレクトリはユーザあたり 25GiBを利用できます。

使用容量は `t4-user-info disk home` コマンドにて確認できます。 以下は、TESTUSERのHOMEディレクトリの容量を確認する例です。

```
[login]$ t4-user-info disk home
  uid name         b_size(GB) b_quota(GB)    i_files    i_quota
-------------------------------------------------------------------
 2011 TESTUSER             7          25     101446    2000000
```

25GBのクォータ制限のうち、7GB利用し、inode制限については、200万のクォータ制限のうち、約10万利用している状況が確認できます。

クォータ制限を超過した場合、新規の書き込みができなくなりますのでご注意ください。 クォータ制限を下回るように容量を削減すれば再度書き込みが可能になります。

稀に容量を削減してもクォータ制限を超過したままの状態が維持される場合があります。 その際は最大　日程度待機する事でクォータの再計算処理が行われ、正常な値に戻ります。

3.2. グループディスク

グループディスクは、グループメンバで共有して使用します。グループディスクは、目的に応じて高速および大容量ストレージ領域を選択することができます。 グループディスクの購入方法は「TSUBAME4.0ポータル利用説明書」のグループディスクの管理をご参照ください。
それぞれのストレージ領域の特徴は以下の通りです。

領域	PATH	グループディスクの容量設定	購入可能な領域	保存可能なファイル数(inode数)
高速ストレージ領域(SSD)	/gs/fs	100GB単位	最大 3TB	2,000,000 / 1TB
大容量ストレージ領域(HDD)	/gs/bs	1TB単位	最大 100TB	2,000,000 / 1TB

3.2.1. 高速ストレージ領域

高速ストレージ領域は/gs/fs にマウントされています。SSDを用いたLustreファイルシステムで構成され、グループディスクとして購入することで利用することができます。
大容量ストレージ領域に比べてディスクの総容量が小さいですが、ファイルアクセスが高速なため「ファイルアクセスの速度が重要な実行環境」「アクセス頻度の高いデータファイル」等の利用を推奨します。

3.2.2. 大容量ストレージ領域

大容量ストレージ領域は/gs/bs にマウントされています。ハードディスクを用いたLustreファイルシステムで構成され、グループディスクとして購入することで利用することができます。
高速ストレージ領域に比べてファイルアクセスは低速ですが、利用できるストレージサイズが大きく利用コストも安いため「ファイルアクセスの速度を求めないデータ」「バックアップデータなどファイルの保管場所」等の利用を推奨します。

3.2.3. グループディスクの使用容量

グループディスクの使用容量はt4-user-info disk group コマンドにて確認できます。 以下は、TESTGROUPのグループディスクの容量を確認する例です。

```
[login]$ t4-user-info disk group -g TESTGROUP
                                        /gs/bs                                  /gs/fs
  gid group_name                 size(TB) quota(TB)    file(M)   quota(M)  size(TB) quota(TB)    file(M)   quota(M)
-------------------------------------------------------------------------------------------------------------------
 xxxx TESTGROUP                    59.78       100       7.50        200      0.00          0      0.00          0
```

指定したTESTGROUPグループでは、/gs/bsのみ購入し、100TBのクォータ制限のうち、約60TB利用し、inode制限については、2億のクォータ制限のうち、750万利用している状況が確認できます。

3.2.4. グループディスクの利用状況確認

グループディスクはひと月ごとに購入します。
前月と今月でグループディスクの購入量が異なる　減少する　場合、利用状況によってはグループディスクの購入量よりも実際の使用量が大きくなる場合があります。
以下のいずれかの手順を用いてグループディスクの使用量が購入量を超えていないか確認してください。
[確認方法]
1. TSUBAME4.0上で確認する場合 : グループディスクの使用容量参照
2. TSUBAMEポータル上で確認する場合 : グループディスクの利用状況確認参照

> **Info**
>
> TSUBAMEポータル上での確認は、グループ管理者及び権限を付与されたグループメンバだけが行うことができます。

3.2.4.1. グループディスクの使用量超過時の対応

グループディスクの使用量が購入量を上回っている場合、以下の2段階の制限がかかります。
1. 購入量以上のファイルの書き込み禁止　月初
2. グループディスク内の一切のアクセス禁止　特定タイミング

1.の状態の場合、ファイルを削除し購入量内に収めるかグループディスクの容量設定を参考にグループディスクの購入サイズを変更し、現在の使用量を超えるようにしてください。
2.の状態の場合、ファイル削除ができません。グループディスクの容量設定を参考にグループディスクの購入サイズを変更し、現在の使用量を超えるようにしてください。

3.3. 学内からのCIFSによるアクセス

TSUBAME4.0では、グループディスクに対して学内のWindows/Mac端末からCIFSによるアクセスが可能です。 以下のアドレスでアクセスすることができます。

```
\\gshs.t4.gsic.titech.ac.jp
```

アカウントはTSUBAME4.0のアカウント、パスワードはポータルで設定したパスワードになります。 Windowsからアクセスする際には、以下のように@TSUBAMEを指定してください。

ユーザー名	*(TSUBAME4.0*アカウント名*)*@TSUBAME
パスワード	*(TSUBAME4.0*アカウントのパスワード*)*

/gs/bs、/gs/fs に対応して、t4_bs、t4_fs となっています。 グループディスクとして購入したディレクトリへアクセスしてください。

Info

ホームディレクトリはCIFSでアクセスできません。
CIFSでの接続に失敗する場合、グループディスクへのCIFS接続ができない, WindowsでTSUBAMEのグループディスクが開けないを参照してください。

3.4. ローカルスクラッチ領域

ローカルスクラッチ領域は計算ノード内のSSD上にあり、処理の一時ファイルや途中結果を高速に読み書きできます。
詳細についてはローカルスクラッチ領域を参照してください。

4. ソフトウェア環境

本ページのコマンドライン例では、以下の表記を使用します。
[login]$: ログインノード
[rNnN]$: 計算ノード
[login/rNnN]$: ログインノードまたは計算ノード
[yourPC]$: ログインノードへの接続元環境

4.1. 利用環境の切換え方法

本システムでは、Environment Modules(moduleコマンド)を使用することでコンパイラやアプリケーション利用環境の切り替えを行うことができます。
Environment Modulesについては man module または公式サイトをご参照ください。

4.1.1. 利用可能なmodule環境の表示

利用可能なmodule環境はmodule availまたはmodule avaで確認できます。

```
[login/rNnN]$ module avail
```

読み込めるバージョンについてはTSUBAME計算サービスWebページのシステム構成の下記項目をご確認下さい。
システムソフトウェア
サポートされているアプリケーション

4.1.2. module環境の設定情報表示

module環境の設定情報を確認したい場合、「module whatisモジュール名」を実行します。

```
[login/rNnN]$ module whatis intel/2024.0.2
------------------------------ /apps/t4/rhel9/modules/modulefiles/compiler -------------------------------
      intel/2024.0.2: Intel oneAPI compiler 2024.0 and MKL
```

4.1.3. module環境のロード

module環境をロードしたい場合、「module load モジュール名」を実行します。

```
[login/rNnN]$ module load intel
```

バッチスクリプトにおいてロードするmoduleは、コンパイル時と同様のものをロードしてください。

4.1.4. module環境の表示

現在使用しているmodule環境を確認したい場合、「module list」を実行します。

```
[login/rNnN]$ module list
Currently Loaded Modulefiles:
 1) intel/2024.0.2   2) cuda/12.3.2
```

4.1.5. module環境のアンロード

ロードしたmodule環境をアンロードしたい場合「module unload モジュール名」を実行します。

```
[login/rNnN]$ module list
Currently Loaded Modulefiles:
 1) intel/2024.0.2   2) cuda/12.3.2
[login/rNnN]$ module unload cuda
[login/rNnN]$ module list
Currently Loaded Modulefiles:
 1) intel/2024.0.2
```

4.1.6. module環境の初期化

ロードしたmodule環境を初期化したい場合、「module purge」を実行します。

```
[login/rNnN]$ module list
Currently Loaded Modulefiles:
 1) intel/2024.0.2   2) cuda/12.3.2
[login/rNnN]$ module purge
[login/rNnN]$ module list
No Modulefiles Currently Loaded.
```

4.2. Intelコンパイラ

本システムではコンパイラとして、Intel コンパイラ (oneAPI) 、AMD コンパイラ (AOCC)、NVIDIA コンパイラ (NVIDIA HPC SDK) および GNU コンパイラが利用できます。

Intel コンパイラの各コマンドは以下のとおりです。

コマンド	言語	コマンド形式	旧コマンド　廃止
ifx	Fortran 77/90/95	`$ ifx [オプション] source_file`	ifort
icx	C	`$ icx [オプション] source_file`	icc
icpx	C++	`$ icpx [オプション] source_file`	icpc

利用する際は、moduleコマンドでintelを読み込んでください。--helpオプションを指定して頂くとコンパイラオプションの一覧が表示されます。

Info

Intel oneAPI 2024から、iccコマンドおよびicpcコマンドが使用できなくなりました。 icxコマンドおよびicpxコマンドを使用してください。
また、Intel oneAPI 2025から、ifortコマンドが使用できなくなりました。
ifxコマンドを使用してください。

Info

Intel oneAPI 2024から、C++のデフォルトの規格がC++14からC++17に変更になりました。それに伴い、一部文法がエラーとなります。詳細については こちら をご参照ください。

4.2.1. コンパイルの主なオプション

コンパイルの最適化オプションを以下に示します。

オプション	説明
`-O`	O2 と同じ。
`-O0`	すべての最適化を無効にします。
`-O1`	最適化を有効にします。コードサイズを大きくするだけで高速化に影響を与えるような一部の最適化を無効にします。
`-O2`	最適化を有効にします。一般的に推奨される最適化レベルです。 ベクトル化は O2 以上のレベルで有効になります。-Oオプションを指定しない場合、デフォルトでこちらが指定されます。
`-O3`	O2 よりも積極的に最適化を行い、融合、アンロールとジャムのブロック、IF 文の折りたたみなど、より強力なループ変換を有効にします。
`-xCORE-AVX512`	Intel プロセッサー向けのIntel アドバンスト・ベクトル・エクステンション 512 (Intel AVX512)、Intel AVX2、AVX、SSE4.2、SSE4.1、SSSE3、SSE3、SSE2、SSE 命令を生成します。Intel AVX512 命令セット対応のIntel プロセッサー向けに最適化します。
`-xCORE-AVX2`	Intel プロセッサー向けのIntel アドバンスト・ベクトル・エクステンション 2 (Intel AVX2)、Intel AVX、SSE4.2、SSE4.1、SSSE3、SSE3、SSE2、SSE 命令を生成します。Intel AVX2 命令セット対応のIntel プロセッサー向けに最適化します。
`-xSSE4.2`	Intel プロセッサー向けのIntel SSE4 高効率および高速な文字列処理命令、Intel SSE4 ベクトル化コンパイラ命令およびメディア・アクセラレーター命令、およびIntel SSSE3、SSE3、SSE2、SSE 命令を生成します。Intel SSE4.2 命令セット対応のIntel プロセッサー向けに最適化します。
`-xSSSE3`	Intel プロセッサー向けのIntel SSSE3、SSE3、SSE2、SSE 命令を生成します。Intel SSSE3 命令セット対応のIntel プロセッサー向けに最適化します。xオプションを指定しない場合、デフォルトでこちらが指定されます。
`-qopt-report=n`	最適化レポートを生成します。デフォルトでは、レポートは.optrpt 拡張子を持つファイルに出力されます。　には、0 (レポートなし) から5 (最も詳しい) の詳細レベルを指定します。デフォルトは 2 です。
`-fp-model precise`	浮動小数点演算のセマンティクスを制御します。浮動小数点データの精度に影響する最適化を無効にし、中間結果をソースで定義された精度まで丸めます。
`-g`	-gオプションはオブジェクト・ファイルのサイズを大きくするシンボリック・デバッグ情報をオブジェクト・ファイルに生成するようにコンパイラに指示します。
`-traceback`	このオプションは、ランタイム時に致命的なエラーが発生したとき、ソースファイルのトレースバック情報を表示できるように、オブジェクト・ファイル内に補足情報を生成するようにコンパイラに指示します。 致命的なエラーが発生すると、コールスタックの 16 進アドレス (プログラム・カウンター・トレース) とともに、ソースファイル、ルーチン名、および行番号の相関情報が表示されます。 マップファイルとエラーが発生したときに表示されるスタックの 16 進アドレスを使用することで、エラーの原因を特定できます。 このオプションを指定すると、実行プログラムのサイズが増えます。

4.2.2. コンパイルの推奨最適化オプション

コンパイルの推奨最適化オプションを以下に示します。本システムに搭載している AMD EPYC 9654 は、Intel AVX512 命令セットに対応していますので、-xCORE-AVX512オプションを指定することができます。-xCORE-AVX512を指定すると、コンパイラがソースコードを解析し、最適なAVX512、AVX2、AVX、SSE命令を生成します。推奨最適化オプションは積極的な最適化を行い、かつ安全なオプションです。最適化のために計算の順序を変更する可能性があり、結果に誤差が生じる場合があります。

Info

Intel Xeon　Skylake 以降　で採用された AVX512 は、AMD では本システムに搭載している Zen4 アーキテクチャの第4世代 EPYC で対応しました。

オプション	説明
`-O3`	O2 最適化を行い、融合、アンロールとジャムのブロック、IF 文の折りたたみなど、より強力なループ変換を有効にします。
`-xCORE-AVX512`	Intel プロセッサー向けのIntel アドバンスト・ベクトル・エクステンション 512 (Intel AVX512)、Intel AVX2、AVX、SSE4.2、SSE4.1、SSSE3、SSE3、SSE2、SSE 命令を生成します。Intel AVX512 命令セット対応のIntel プロセッサー向けに最適化します。

上記のオプションを使用することにより、プログラムの性能が悪化した場合、最適化のレベルを-O2に下げるかベクトル化のオプションを変更してください。 また、結果が一致していない場合、浮動小数点のオプションも試してみてください。

4.2.3. Intel 64アーキテクチャーのメモリモデル指定

次のいずれかのメモリモデルを使用して実行バイナリを作成します。

メモリモデル	説明
small (`-mcmodel=small`)	コードとデータのすべてのアクセスが、命令ポインター (IP) 相対アドレス指定で行われるように、コードとデータはアドレス空間の最初の 2GB までに制限されます。 -mcmodelオプションを指定しない場合、デフォルトでこちらが指定されます。
medium (`-mcmodel=medium`)	コードはアドレス空間の最初の 2GB までに制限されますが、データは制限されません。コードは IP 相対アドレス指定でアクセスできますが、データのアクセスは絶対アドレス指定を使用する必要があります。
large (`-mcmodel=large`)	コードもデータも制限されません。コードもデータもアクセスは絶対アドレス指定を使用します。

IP 相対アドレス指定は 32 ビットのみ必要ですが、絶対アドレス指定は 64 ビット必要です。 これは、コードサイズとパフォーマンスに影響します。 (IP 相対アドレス指定の方が多少速くアクセスできます。)

プログラム内の共通ブロック、グローバルデータ、静的データの合計が2GBを越えるとき、リンク時に次のエラーメッセージが出力されます。

```
<some lib.a library>(some .o): In Function <function>:
  : relocation truncated to fit: R_X86_64_PC32 <some symbol>
.....
  : relocation truncated to fit: R_X86_64_PC32 <some symbol>
```

この場合は、`-mcmodel=medium` と `-shared-intel` を指定してコンパイル/リンクして下さい。 medium メモリモデルまたは large メモリモデルを指定した場合、Intelのランタイム・ライブラリの適切なダイナミック・バージョンが使用されるように、-shared-intel コンパイラ・オプションも指定する必要があります。

4.3. NVIDIA HPC SDK

NVIDIA HPC SDK　旧PGIコンパイラ　の各コマンドは以下のとおりです。

コマンド	言語	コマンド形式	旧コマンド
nvfortran	Fortran 77/90/95/2003/2008/2018	`$ nvfortran [オプション] source_file`	pgfortran
nvc	C	`$ nvcc [オプション] source_file`	pgcc
nvc++	C++	`$ nvc++ [オプション] source_file`	pgc++

各コマンドの詳細は `$ man nvcc` 等でご確認下さい。

利用する際は、moduleコマンドでnvhpcを読み込んでください。--helpオプションを指定して頂くとコンパイラオプションの一覧が表示されます。

4.4. AOCC

AMD Optimizing C/C++ and Fortran Compilers (AOCC) の各コマンドは以下のとおりです。

コマンド	言語	コマンド形式
flang (clang)	Fortran 95/2003/2008	`$ flang [オプション] source_file`
clang	C	`$ clang [オプション] source_file`
clang-cpp (clang)	C++	`$ clang-cpp [オプション] source_file`

利用する際は、moduleコマンドでaoccを読み込んでください。--helpオプションを指定して頂くとコンパイラオプションの一覧が表示されます。

4.5. GPU環境

本システムではGPU　NVIDIA H100 SXM5　の利用環境を提供しております。

4.5.1. インタラクティブジョブの実行・デバッグ

ログインノード　login, login1, login2　には、GPUを搭載しておらず、コンパイル、リンクのみ実行可能です。 また、ログインノードにおける高負荷プログラムの実行は制限されています。

インタラクティブでの実行、デバックについては、バッチシステムを使用して実行可能です。 詳細については、インタラクティブジョブの投入を参照ください。

4.5.2. 対応アプリケーション

現在のGPU対応アプリケーションは次の通りです。(2024.4.1現在

- ABAQUS 2024 --- ABAQUS利用の手引(別冊) を参照ください。
- ANSYS 2024R1 --- ANSYS 利用の手引(別冊) を参照ください。
- AMBER 22up05 --- AMBER利用の手引(別冊) を参照ください。
- Mathematica 14 --- Mathematica利用の手引(別冊) を参照ください。
- MATLAB 2024 --- MATLAB利用の手引(別冊) を参照ください。
- Linaro forge(旧:Arm Forge) --- 講習会内の「並列プログラミング」を参照ください。
- NVIDIA HPC SDK --- NVIDIA コンパイラ 利用の手引(別冊) を参照ください。

他のアプリケーションにつきまても、順次展開してまいります。

4.5.3. CUDA 対応のMPI

CUDA版に対応したMPI環境を用意しております。

OpenMPI + gcc環境

```
# OpenMPI, GCC環境の読込
[rNnN]$ module load openmpi/5.0.2-gcc
Loading openmpi/5.0.2-gcc
  Loading requirement: cuda/12.3.2
```

必要な関連 module は自動でロードされます。

OpenMPI + NVIDIA HPC SDK環境

```
# OpenMPI、NVIDIA HCP SDK 環境の読込
[rNnN]$ module load openmpi/5.0.2-nvhpc
Loading openmpi/5.0.2-nvhpc
```

OpenMPI + Intel環境

```
# OpenMPI, Intel環境の読込
[rNnN]$ module load openmpi/5.0.2-intel
Loading openmpi/5.0.2-intel
  Loading requirement: intel/2024.0.2 cuda/12.3.2
```

必要な関連 module は自動でロードされます。

4.5.4. Multi-Instance GPU (MIG)

資源タイプ node_o および gpu_h では、GPU数 1/2 が割り当てられています。
これはMulti-Instance GPU(MIG)機能を使用して、TSUBAME4.0の計算ノードに搭載されているNVIDIA H100 SXM5 94GB HBM2eを2つに論理分割することで実装しています。
MIGの詳細についてはこちらを参照してください。
なお、その他の資源タイプでは、MIG機能は使用しておりません。

4.5.5. Multi-Process Service (MPS)

マルチプロセスサービス (MPS) を使用すると、単一の GPU で複数の CUDA プロセスを使用できます。
プロセスは GPU 上で並行して実行されるため、GPU コンピュートリソースの飽和が発生しなくなります。
MPS を使用すると、カーネル操作や、別のプロセスからのメモリーコピーの同時実行または重複も可能になります。
MPSの詳細についてはこちらをご参照ください。

TSUBAME4.0でMPSを利用した際に、同一ノード上で実行されている別のジョブに障害が発生する事案が発生しました。
本障害を回避するため、TSUBAME4.0では独自の nvidia-cuda-mps-control コマンドを用意しています。
module load cuda
を実行することで、利用できるようになります。必ずこちらのコマンドを使用してください。

また、MPS利用時に環境変数 CUDA_MPS_PIPE_DIRECTORY を設定するよう紹介されているサイトがありますが、**TSUBAME4.0**では当該環境変数を変更してはいけません。
利用者が独自にCUDA_MPS_PIPE_DIRECTORY を設定することで、同様に上記障害が発生することがわかっています。

これらのルールを守らず他の利用者に損害を与えた場合、無予告でのジョブ削除や計算機アカウントの一時停止などの措置を行う可能性もありますので十分ご注意ください。

4.5.6. GPUのCOMPUTE MODEの変更

資源タイプnode_fを利用した場合、GPUのCOMPUTE MODEを変更することが出来ます。 GPUのCOMPUTE MODEを変更するには、node_fを指定した上で、ジョブスクリプトの中で、以下を指定してください。

```
#$ -v GPU_COMPUTE_MODE=<利用するモードの番号>
```

利用可能なモードは以下の3つです。

番号	モード	説明
0	DEFAULTモード	1つのGPUを複数のプロセスから同時に利用できる。
1	EXCLUSIVE_PROCESSモード	1つのGPUを1プロセスのみが利用できる。1プロセスから複数スレッドの利用は可能。
2	PROHIBITEDモード	GPUへのプロセス割り当てを禁止する。

node_f指定時にGPU_COMPUTE_MODEを指定しない場合、DEFAULTモード(0)が設定されます。
また、資源タイプにnode_f以外を指定した場合、GPU_COMPUTE_MODEはDEFAULTモード(0)固定です。

以下はスクリプトの例となります。

```
#!/bin/sh
#$ -cwd
#$ -l node_f=1
#$ -l h_rt=1:00:00
#$ -N gpumode
#$ -v GPU_COMPUTE_MODE=1
/usr/bin/nvidia-smi
```

インタラクティブで利用する場合、qrshは以下のような形となります。

```
[login]$ qrsh -g [TSUBAMEグループ] -l node_f=1 -l h_rt=0:10:00 -pty yes -v TERM -v GPU_COMPUTE_MODE=1 /bin/bash
```

4.6. コンテナの利用

TSUBAME4.0では、HPC向けコンテナ環境として Apptainer(旧:Singularity) が利用可能です。

Apptainer の使い方の例を以下に記します。

4.6.1. イメージの作成

Apptainer のイメージ作成手順例について以下に示します。ここでは、Ubuntu の最新 Docker イメージを使用します。

[オプション]

- -nv : GPUを使用する
- -B : ファイルシステムをマウントする
- -s : サンドボックス形式でイメージをビルドする

```
[login]$ mkdir $HOME/apptainer
[login]$ cd $HOME/apptainer
[login]$ apptainer build -s ubuntu/ docker://ubuntu:latest
INFO:    Starting build...
Getting image source signatures
Copying blob 49b384cc7b4a done
Copying config bf3dc08bfe done
Writing manifest to image destination
Storing signatures
2024/05/29 13:07:49  info unpack layer: sha256:49b384cc7b4aa0dfd16ff7817ad0ea04f1d0a8072e62114efcd99119f8ceb9ed
```

```
2024/05/29 13:07:50  warn xattr{etc/gshadow} ignoring ENOTSUP on setxattr "user.rootlesscontainers"
2024/05/29 13:07:50  warn xattr{$HOME/apptainer/build-temp-2088960457/rootfs/etc/gshadow} destination filesystem does not support xattrs, further warnings
will be suppressed
INFO:    Creating sandbox directory...
INFO:    Build complete: ubuntu/

[login apptainer]$ cd ubuntu/
[login apptainer]$ mkdir gs apps      # /gs /apps をマウントするためのディレクトリを作成します
[login apptainer]$ cd ../
```

4.6.2. シェルの起動

Apptainer でシェルを起動する手順例について以下に示します。 ここでは、イメージの作成 で作成したイメージを使用します。

[オプション]

- -nv : GPUを使用する
- -B : ファイルシステムをマウントする
- -f (--fakeroot) : コンテナ内でroot権限を行使する
- -w (--writable) : コンテナ内への書き込みを可能にする

> **Info**
>
> -w (--writable) オプションは必ず指定してください。
> 指定しない場合、正常に起動しないことを確認しています。

```
[login]$ cd $HOME/apptainer
[login]$ apptainer shell -B /gs -B /apps -B /home --nv -f -w ubuntu/
INFO:    User not listed in /etc/subuid, trying root-mapped namespace
INFO:    Using fakeroot command combined with root-mapped namespace
WARNING: nv files may not be bound with --writable
WARNING: Skipping mount /etc/localtime [binds]: /etc/localtime doesn't exist in container
WARNING: Skipping mount /bin/nvidia-smi [files]: /usr/bin/nvidia-smi doesn't exist in container
WARNING: Skipping mount /bin/nvidia-debugdump [files]: /usr/bin/nvidia-debugdump doesn't exist in container
WARNING: Skipping mount /bin/nvidia-persistenced [files]: /usr/bin/nvidia-persistenced doesn't exist in container
WARNING: Skipping mount /bin/nvidia-cuda-mps-control [files]: /usr/bin/nvidia-cuda-mps-control doesn't exist in container
WARNING: Skipping mount /bin/nvidia-cuda-mps-server [files]: /usr/bin/nvidia-cuda-mps-server doesn't exist in container
Apptainer> id
uid=0(root) gid=0(root) groups=0(root)
```

4.6.2.1. fakeroot利用時の留意事項について

Apptainer でフェイクルート機能(--fakeroot)を使用する場合、Apptainer はホスト側のfakerootをバインドマップして利用するためホストとコンテナの間でlibcのバージョンを一致させる必要があります。
ホストの libc ライブラリがコンテナー内の対応するライブラリよりも新しい場合、fakeroot コマンドは GLIBC バージョンが不足しているというエラーを出力する場合があります。(参考:Fakeroot feature)
エラー出力例

```
/.singularity.d/libs/faked: /lib/x86_64-linux-gnu/libc.so.6: version`GLIBC_2.33' not found (required by /.singularity.d/libs/faked)
/.singularity.d/libs/faked: /lib/x86_64-linux-gnu/libc.so.6: version`GLIBC_2.34' not found (required by /.singularity.d/libs/faked)
fakeroot: error while starting the `faked' daemon.
/.singularity.d/libs/fakeroot: 1: kill: Usage: kill [-s sigspec | -signum |-sigspec] [pid | job]... or
kill -l [exitstatus]
```

上記のようなエラーが発生した場合、ホストとコンテナの libc ライブラリバージョンが一致するようにコンテナを再作成してください。
また、実験的サービスのため未サポートですが、本問題を回避するための互換ライブラリを用意しています。

5. ジョブスケジューリングシステム

本ページのコマンドライン例では、以下の表記を使用します。
[login]$: ログインノード
[rNnN]$: 計算ノード
[login/rNnN]$: ログインノードまたは計算ノード
[yourPC]$: ログインノードへの接続元環境

本システムのジョブスケジューリングには、シングルジョブ・並列ジョブを優先度や必要なリソースに従い効率的にスケジューリングする、「Altair Grid Engine」を採用しています。

5.1. ジョブスケジューリングシステムの構成

TSUBAME4.0では、用途に応じて以下のキュー種別 / ジョブ種別を用意しています。

キュー種別	ジョブ種別	コマンド	資源タイプ
通常キュー	バッチジョブ	qsub	指定可能
	インタラクティブジョブ	qrsh	指定可能
インタラクティブジョブ専用キュー	インタラクティブジョブ	iqrsh	固定

[キュー種別]

・通常キュー
論理的に分割した資源タイプ単位で、システムリソースを確保・占有利用します。 通常はこちらを使用します。ノード予約も利用可能です。

・インタラクティブジョブ専用キュー
リソースをユーザ間で共有することで、混雑時にもノード確保に失敗しにくく、可視化や対話的なプログラムを素早く開始できるように用意された環境です。
プロセッサの利用が間欠的な対話型プログラムの実行を想定しています。連続的にプロセッサを占有し続ける計算には使用しないでください。
利用にあたっては事前に各種制限値一覧を必ずお読みください。

インタラクティブジョブ専用キューは、学内ユーザ(tgz-edu)とアクセスカードユーザに限り無償で実行可能です。 無償利用方法についてはインタラクティブジョブ専用キューを参照してください。

[ジョブ種別]

・バッチジョブ
ジョブスクリプトを作成・投入します。詳細についてはバッチジョブをご参照ください。

・インタラクティブジョブ
インタラクティブにプログラムやシェルスクリプトを実行します。キュー種別により利用方法が異なります。
「通常キュー」利用時は、インタラクティブジョブをご参照ください。
「インタラクティブジョブ専用キュー」利用時は、インタラクティブジョブ専用キューをご参照ください。

[資源タイプ]

・通常キュー利用時
計算ノードを論理的に分割した資源タイプを利用して、システムリソースを確保します。利用可能な資源タイプについては、利用可能な資源タイプをご参照ください。

・インタラクティブジョブ専用キュー利用時
物理CPUコア数 24コア, 96GBメモリ, 1MIG(GPU数 1/2)の資源を最大**12**名で共有利用します。

5.1.1. 利用可能な資源タイプ

本システムでは計算ノードを論理的に分割した資源タイプを利用して、システムリソースを確保します。
ジョブ投入の際には、資源タイプをいくつ使うかを指定します(例 `-l node_f=2`)。利用できる資源タイプの一覧を以下に示します。

資源タイプ	使用物理CPUコア数	メモリ (GB)	GPU 数	ローカルスクラッチ領域 (GiB)
node_f	192	768	4	1660
node_h	96	384	2	830
node_q	48	192	1	415
node_o	24	96	1/2	200
gpu_1	8	96	1	200
gpu_h	4	48	1/2	100
cpu_160	160	368	0	830
cpu_80	80	184	0	415
cpu_40	40	92	0	200
cpu_16	16	36.8	0	83
cpu_8	8	18.4	0	40
cpu_4	4	9.2	0	20
インタラクティブジョブ専用キュー利用時(*)	24	96	1/2	200

・「使用物理CPUコア数」、「メモリ(GB)」、「GPU数」は、各資源タイプ1つあたりの使用可能な量です。

・[資源タイプ]=[個数]で同じ資源タイプを複数指定できます。資源タイプの組み合わせはできません。

・実行可能時間の最大値は24時間です。

・*インタラクティブジョブ専用キュー利用時は、確保したシステムリソースを最大12人で共有利用します。全て利用できるわけではありません。

5.1.2. ジョブスケジューラ関連の制限値

TSUBAME4では「同時に実行可能なジョブ数」「実行可能な総スロット数」「1ユーザあたり同時に投入可能なジョブ数」など各種制限値があります。
(スロット=資源タイプ毎に設定されている物理CPUコア数x利用ノード数(qstatコマンドのslotsと同等))

現在の制限値の一覧は以下のURLで確認できます。

https://www.t4.cii.isct.ac.jp/resource-limit

その他、以下の注意事項があります。

・「1ユーザあたり同時に投入可能なジョブ数」は実行中および実行待ちのジョブが対象です。実行が終了したジョブは含まれません。
また、制限値に達した場合、qsub等のコマンド実行時にエラーとなります。詳細についてはジョブ投入時にエラーになりますが、どのオプションが悪いかわかりませんをご確認ください。

・「1ユーザあたり同時に投入可能なジョブ数」はシステムダウンを防ぐための設定であり、この数までであれば投入してよいという意味ではありません。 可能な範囲でジョブをまとめるなど、スケジューラ負荷軽減にご協力ください。

・各種制限値は、利用状況に応じて随時変更する可能性があります。

5.2. 通常キュー

5.2.1. バッチジョブ

本システムでバッチジョブを実行するには、ログインノードへログインしてqsubコマンドを実行します。

5.2.1.1. バッチジョブの流れ

ジョブを投入するためにはジョブスクリプトを作成し投入します。 または、コマンドラインにキュー名などを指定してジョブを投入することもできます。 投入コマンドは"qsub"です。

・ジョブスクリプトの作成

・qsubを使用しジョブを投入

・qstatなどを使用しジョブの状態確認

・必要に応じてqdelを使用しジョブのキャンセル

・ジョブの結果確認

qsubコマンドは、課金情報(TSUBAMEポイント)を確認し、ジョブを受け付けます。

5.2.1.2. ジョブスクリプト

ジョブスクリプトの記述方法を以下に示します。

```
#!/bin/sh
#$ -cwd
#$ -l [資源タイプ] =[個数]
#$ -l h_rt=[経過時間]
#$ -p [プライオリティ]

# TSUBAME4.0 では不要になりました。
# . /etc/profile.d/modules.sh  # moduleの初期化

[プログラミング環境のロード]

[プログラム実行]
```

Warning

shebang(`#!/bin/sh`の箇所)は必ずジョブスクリプトの先頭に来るようにして下さい。

・[プログラミング環境のロード]
moduleコマンドを用い、必要な環境のロードを行います。
intelコンパイラをロードする場合の例は以下となります。

```
[login/rNnN]$ module load intel
```

・[プログラム実行]
プログラムの実行を行います。
バイナリがa.outの場合の例は以下となります。

```
[login/rNnN]$ ./a.out
```

資源タイプの指定などはコマンドラインで指定するか、またはスクリプトファイルの最初のコメントブロック(`#$`)に記述することで有効になります。 資源タイプ、実行時間は必須項目になるため必ず指定するようにしてください。

qsubコマンドの主なオプションを以下に示します。

オプション	説明
-l [資源タイプ]=[個数] (必須)	資源タイプおよびその個数を指定します。
-l h_rt=[経過時間] (必須)	Wall time(経過時間)を指定します。[[HH:]MM:]SSで指定することができます。HH:MM:SやMM:SSやSSのように指定することができます。
-N	ジョブ名を指定します。(指定しない場合はスクリプトファイル名)
-o	標準出力ファイル名を指定します。
-e	標準エラー出力ファイル名を指定します。
-j y	標準エラー出力を標準出力ファイルに統合します。
-m	ジョブについての情報をメールで送信する条件を指定します。 a バッチシステムによりジョブが中止された場合 b ジョブの実行が開始された場合 e ジョブの実行が終了した場合 abeのように組み合わせることも可能です。 メールオプションをつけて大量のジョブを投入すると、大量のメールによってメールサーバーに負荷が掛かり、攻撃と検知され他の利用者もまとめて東京科学大からのメールを遮断される可能性があります。そのようなジョブを流す必要がある場合は、メールオプションを外すか一度のジョブで実行できるようスクリプトの見直しを行ってください。
-p　プレミアムオプション)	ジョブの実行優先度を指定します。 -3,-4を指定すると通常よりも高い課金係数が適用されます。設定値の-5,-4,-3は課金規則の優先度0,1,2に対応します。 -5 : 標準の実行優先度です。 (デフォルト) -4 : 実行優先度は-5より高く,-3より低くなります。 -3 : 最高の実行優先度となります。 優先度の値は全て負の数です。マイナス記号を含めて指定してください。
-t	タスクIDの範囲を指定します。 開始番号-終了番号[:ステップサイズ] で指定することができます。
-hold_jid	依存関係にあるジョブIDを指定します。 指定された依存ジョブが終了しなければ、発行ジョブは実行されません。
-ar	予約ノードを利用する際に 予約AR IDを指定します。
-V	ジョブ投入環境で指定された環境変数を実行環境に渡します。 TSUBAME固有の制限事項として、LD_LIBRARY_PATH、LD_PRELOADなど一部環境変数の値は渡すことが出来ません。

5.2.1.3. ジョブスクリプトの記述例

5.2.1.3.1. シングルジョブ/GPUジョブ

シングルジョブ(並列化されていないジョブ)を実行する時に作成するバッチスクリプトの例を以下に示します。GPUを使用するジョブの場合は `-l cpu_4=1` を `-l gpu_1=1` に変更し、GPUで利用するmoduleの読み込み以外はシングルジョブと同様になります。

```
#!/bin/sh
#$ -cwd          # カレントディレクトリでジョブを実行する場合に指定
#$ -l cpu_4=1
#$ -l h_rt=1:00:00    # 実行時間を指定
#$ -N serial

module load cuda      # CUDA環境の読込
module load intel     # Intel Compiler環境の読込
./a.out
```

5.2.1.3.2. SMP並列

SMP並列ジョブを実行する時に作成するバッチスクリプトの例を以下に示します。計算ノードはハイパースレッディングが有効になっています。使用するスレッド数につきましては、明示的に指定してください。

```
#!/bin/sh
#$ -cwd
#$ -l node_f=1                # 資源タイプnode_f 1ノードを使用
#$ -l h_rt=1:00:00
#$ -N openmp

module load cuda
module load intel
export OMP_NUM_THREADS=192    # ノード内に192スレッドを配置
./a.out
```

5.2.1.3.3. MPI並列

MPI並列ジョブを実行する時に作成するバッチスクリプトの例を以下に示します。使用するMPI環境により使い分けをお願いします。OpenMPIでスレーブノードにライブラリ環境変数を渡たすには、`-x LD_LIBRARY_PATH`を利用する必要があります。

Intel MPI環境

```
#!/bin/sh
#$ -cwd
#$ -l node_f=4                        # 資源タイプnode_f 4ノードを使用
#$ -l h_rt=1:00:00
#$ -N flatmpi

module load cuda
module load intel
module load intel-mpi                  # Intel MPI環境の読込
mpiexec.hydra -ppn 8 -n 32 ./a.out     # ノードあたり8プロセスMPI全32 プロセスを使用
```

OpenMPI環境

```
#!/bin/sh
#$ -cwd
#$ -l node_f=4                                        # 資源タイプnode_f 4ノードを使用
#$ -l h_rt=1:00:00
#$ -N flatmpi

module load openmpi/5.0.2-intel                       # Open MPI環境の読込: Intelコンパイラ, CUDAが自動でロードされる
mpirun -npernode 8 -n 32 -x LD_LIBRARY_PATH ./a.out   # ノードあたり8プロセスMPI全32 プロセスを使用
```

※ 投入したジョブに対して割り当てられているノードリストは、PE_HOSTFILE変数で参照できます。

```
[rNnN]$ echo $PE_HOSTFILE
/var/spool/age/r15n10/active_jobs/1407687.1/pe_hostfile
[rNnN]$ cat /var/spool/age/r15n10/active_jobs/1407687.1/pe_hostfile
r15n10 24 all.q@r15n10 <NULL>
r20n11 24 all.q@r20n11 <NULL>
r20n10 24 all.q@r20n10 <NULL>
r23n9 24 all.q@r23n9 <NULL>
```

5.2.1.3.4. プロセス並列/スレッド並列(ハイブリッド, MPI+OPENMP)

プロセス並列/スレッド並列(ハイブリッド, MPI+OpenMP)のジョブを実行する時に作成するバッチスクリプトの例を以下に示します。 使用するMPI環境により使い分けをお願いします。 OpenMPIでスレーブノードにライブラリ環境変数を渡たすには、`-x LD_LIBRARY_PATH`を利用する必要があります。

Intel MPI環境

```
#!/bin/sh
#$ -cwd
#$ -l node_f=4                      # 資源タイプnode_f 4ノードを使用
#$ -l h_rt=1:00:00
#$ -N hybrid

module load cuda
module load intel
module load intel-mpi
export OMP_NUM_THREADS=192          # ノード内に192スレッドを配置
mpiexec.hydra -ppn 1 -n 4 ./a.out   # ノードあたりMPI 1プロセス、全4 プロセスを使用
```

OpenMPI環境

```
#!/bin/sh
#$ -cwd
#$ -l node_f=4                                  # 資源タイプF 4ノードを使用
#$ -l h_rt=1:00:00
#$ -N hybrid

module load openmpi/5.0.2-intel                 # Open MPI環境の読込: Intel1コンパイラ, CUDAが自動でロードされる
export OMP_NUM_THREADS=192                      # ノード内に192スレッドを配置
mpirun -npernode 1 -n 4 -x LD_LIBRARY_PATH ./a.out  # ノードあたりMPI 1プロセス、全4 プロセスを使用
```

5.2.1.4. ジョブの投入

ジョブを実行するために、バッチリクエストを事前に作成する必要があります。 qsubコマンドにジョブ投入スクリプトを指定することで、ジョブがキューイングされ実行されます。 qsubコマンドを使用してジョブを投入する例を以下に示します。

```
[login]$ qsub -g [TSUBAMEグループ] スクリプト名
```

オプション	説明
-g	TSUBAMEグループ名を指定します。 スクリプトの中ではなくqsubコマンドのオプションとしてつけてください。
-q prior	定額制ジョブ 実行までに最大1時間の待ちが発生します。

5.2.1.5. ジョブの状態確認

qstatコマンドはジョブ状態表示コマンドです。

```
[login]$ qstat [オプション]
```

qstatコマンドの主なオプションを以下に示します。

qstatコマンドのオプション

オプション	説明
-r	ジョブのリソース情報を表示します。
-j [ジョブID]	ジョブに関する追加情報を表示します。

qstatコマンドの実行結果の例を以下に示します。

```
[login]$ qstat
job-IDprior   nameuser    statesubmit/start at   queuejclass   slotsja-task-ID
-----------------------------------------------------------------------------
307 0.55500 sample.sh testuser r 02/12/2023 17:48:10  all.q@r8i6n1A.default32
(以下省略)
```

qstatコマンドの表示内容を以下に示します。

表示項目	説明
Job-ID	ジョブIDを表示します。
prior	優先度を表示します。
name	ジョブ名を表示します。
user	ジョブのオーナーを表示します。
state	ジョブのステータスを表示します。 r 実行中 qw 待機中 h ホールド中 d 削除中 t 移動中 s サスペンド状態、一時停止 S サスペンド状態、キューのサスペンド状態 T サスペンド状態、制限超過によるサスペンド E エラー状態 Rq 再スケジューリングされ待機中のジョブ Rr 再スケジューリングされ実行中のジョブ
submit/start at	投入/開始日時を表示します。
queue	キュー名を表示します。
jclass	ジョブクラス名を表示します。
slots	利用しているスロット数を表示します。 (スロット=資源タイプ毎に設定されている物理CPUコア数x利用ノード数)
ja-task-ID	アレイジョブに関してタスクIDを表示します。

5.2.1.6. ジョブの削除

バッチジョブを削除する場合には、 `qdel` コマンドを使用します。

```
[login]$ qdel [ジョブID]
```

以下にジョブをqdelした結果を示します。

```
[login]$ qstat
job-IDprior   nameuser    statesubmit/start at   queuejclass   slotsja-task-ID
-----------------------------------------------------------------------------------------
307 0.55500 sample.sh testuser r 02/12/2023 17:48:10   all.q@r8i6n1A.default32

[login]$ qdel 307
testuser has registered the job 307 for deletion

[login]$ qstat
job-IDprior   nameuser    statesubmit/start at   queuejclass   slotsja-task-ID
-----------------------------------------------------------------------------------------
```

5.2.1.7. ジョブの結果確認

AGEのジョブの標準出力はジョブを実行したディレクトリの「スクリプトファイル名.oジョブID」というファイルに保管されます。 また、標準エラー出力は「スクリプトファイル名.eジョブID」です。

5.2.1.8. アレイジョブ

ジョブスクリプト内に含まれる操作をパラメータ化して繰り返し実行する機能としてアレイジョブ機能があります。 アレイジョブで実行される各ジョブをタスクと呼び、タスクIDによって管理されます。 またタスクIDを指定しないジョブIDは、タスクID全部を範囲とします。

Info

アレイジョブの各タスクはそれぞれ別のジョブとしてスケジュールされるため、タスクの数に比例したスケジュールの待ち時間が発生します。
各タスクの処理が短い場合や、タスク数が多い場合は、複数のタスクをまとめてタスクの数を減らすことを強くお勧めいたします。
例: 10000タスクを、それぞれ100タスク分の処理をするタスク100個にまとめる

タスク番号の指定は、qsubコマンドのオプションもしくはジョブスクリプト内で定義します。 投入オプションは `-t (開始番号)-(終了番号):(ステップサイズ)` として指定します。 ステップサイズが1の場合は省略可能です。以下に例を示します。

```
# ジョブスクリプト内にて以下を指定
#$ -t 2-10:2
```

上記例　2-10:2　では、開始番号 2、終了番号 10、ステップサイズ2 (1つ飛ばしのインデックス)が指定され、タスク番号 2、4、6、8、10 の 5 つのタスクによって構成されます。

各タスクのタスク番号は $SGE_TASK_IDという環境変数に設定されるため、この環境変数をジョブスクリプト内で利用することで、パラメータスタディが可能となります。 結果ファイルはジョブ名の後ろにタスクIDが付いた形で出力されます。

また、実行前/実行中に特定のタスクIDを削除したい場合には、以下のように qdelの-tオプションを使用します。

```
[login]$ qdel [ジョブID] -t [タスクID]
```

5.2.2. インタラクティブジョブ

本システムのジョブスケジューラでは、インタラクティブにプログラムやシェルスクリプトを実行する機能を有しています。
インタラクティブジョブを実行するためには、qrshコマンドを使用し、-lで資源タイプ、経過時間を指定します。
qrshでジョブ投入後、ジョブがディスパッチされるとコマンドプロンプトが返ってきます。
インタラクティブジョブの使用方法の流れ以下に示します。

```
[login]$ qrsh -g [TSUBAMEグループ] -l [資源タイプ]=[個数] -l h_rt=[経過時間]
Directory: /home/N/username
(ジョブ開始時刻)
[rNnN]$ [計算ノードで実行したいコマンド]
[rNnN]$ exit
```

-gオプションのグループ指定が未指定の場合は資源タイプ2つまで、経過時間10分間まで、優先度-5の「お試し実行」となります。

資源タイプ node_f、1ノード、経過時間 10分を指定した例

```
[login]$ qrsh -g [TSUBAMEグループ] -l node_f=1 -l h_rt=0:10:00
Directory: /home/N/username
(ジョブ開始時刻)
[rNnN]$ [計算ノードで実行したいコマンド]
[rNnN]$ exit
```

プロンプトにexitと入力することでインタラクティブジョブを終了します。

5.2.2.1. インタラクティブノードを利用したX転送

qrshで接続したノードから直接X転送を行う場合は、下記の手順にて接続ください。

1. X転送を有効にしてログインノードにssh
2. qrshコマンドの実行

コマンド実行例

例では資源タイプcpu_4、1ノードで2時間のジョブを実行しています。

割り当てノードはコマンド実行時に空いているノードですので、明示的にノードを指定することはできません。

```
# qrshの実行
[login]$ qrsh -g [TSUBAMEグループ] -l cpu_4=1 -l h_rt=2:00:00
[rNnN]$ module load [読み込みたいアプリケーション]
[rNnN]$ [実行したいアプリケーションの実行コマンド]
[rNnN]$ exit
```

Info

インタラクティブノードを利用したX転送については、Open OnDemandも利用可能です。

5.2.2.2. ネットワーク系アプリケーションへの接続

Webブラウザ等でアプリケーションを操作する必要がある場合、SSHポートフォワードを用いて手元のWebブラウザからアクセスすることが可能です。

(1) qrshで接続したインタラクティブノードのホスト名の取得

```
[login]$ qrsh -g tsubamegroup00 -l cpu_4=1 -l h_rt=0:10:00
[rNnN]$ hostname
r1n1
[rNnN]$ [Webブラウザ等からのアクセスが必要なプログラムの実行]
```

qrshでインタラクティブジョブを起動後、そのマシンのホスト名を取得します。 上記の例では、ホスト名として `r1n1` がホスト名になります。 この、コンソールでの作業はおわりですが、アプリケーションによる作業が終了するまで、そのまま維持してください。

(2) ssh接続元のコンソールよりSSHのポートフォワードを有効にして接続する。(ログインノードやインタラクティブジョブ上ではありません)

```
[yourPC]$ ssh -i /path/to/key -l username -L 8888:<マシンのホスト名>:<接続するアプリケーションのネットワークポート> login.t4.gsic.titech.ac.jp
```

接続するアプリケーションのネットワークポートは、アプリケーションごとに異なります。詳しくは、各アプリケーションの説明書もしくは、アプリケーションの起動メッセージをご確認ください。

Tips

TSUBAME4にSSHするコンソールによっては、SSHのポートフォワードの設定が異なます。 詳しくは、各SSHコンソールの説明をご確認いただくか、FAQをご参照ください。

(3) Webブラウザでアプリケーションに接続する。 手元のコンソール上でWebブラウザ (Microsoft Edge, Firefox, Safari等)を立ち上げ、http://localhost:8888/にアクセスしてください。

5.2.3. お試し実行

Info

本機能は既にアカウントをお持ちの方(おもに自由にアカウント作成ができる学内利用者)向けの機能です。

TSUBAMEを自分の研究に利用できるか不安のある利用者がポイント購入をする前に動作確認できるよう、TSUBAMEではポイントを消費することなくプログラムの動作確認を行うことができるお試し実行機能が用意されています。
ジョブ投入時に `-g` オプションでグループを指定しないことで、ジョブをお試し実行として投入することができます。 この際、2並列以下、実行時間10分以下、優先度-5(最低) という制限がかかります。

> **Warning**
>
> お試し実行は課金前のプログラムの動作確認を目的とした利用に限り、実際の研究や計測を目的とした実行は行わないでください。
> 制限内でなら無償で無制限に利用してよいというわけではありません。

お試し実行機能はTSUBAMEを自分の研究に利用できるか不安のある利用者がポイント購入をする前に動作確認できるように用意されたものですので、これらの目的から大きく逸脱する、直接研究成果につながるような計算はお試し実行機能で行わないようお願いします。
授業において、教育目的の小規模な計算を実行したい場合はインタラクティブジョブ専用キューをご利用することもご検討ください。

お試し実行の場合、資源量に以下の制限が適用されます。

利用可能な最大ノード数(資源数)	2
利用最長時間	10分
同時実行数	1
資源タイプ	制限なし

また、お試し実行には「TSUBAMEグループ」を指定しないで実行する必要があります。
TSUBAMEグループを指定した場合=-gオプションを利用した場合はポイントが消費されますのでご注意下さい。

5.2.4. 定額制ジョブ

定額制でのジョブ投入は、-q prior をつけます。その他のオプションは従量制と同じです。

```
[login]$ qsub -q prior -g [TSUBAMEグループ] スクリプト名
```

定額制についての詳細はこちらをご覧下さい。

> **Warning**
>
> 定額制グループのジョブであっても、 -q prior の指定がない場合、従量制ジョブとして処理されますので、ご注意ください。

5.2.5. 計算ノードの予約

計算ノードを予約することにより、24時間および72ノードを越えるジョブの実行が可能です。 予約実行の流れは以下のようになります。

- TSUBAMEポータルから予約の実行
- TSUBAMEポータルから予約状況の確認、キャンセル
- 予約ノードに対してqsubを使用しジョブを投入
- 必要に応じてqdelを使用しジョブのキャンセル
- ジョブの結果確認
- コマンドラインからの予約状況およびAR IDの確認

ポータルからの予約の実行、予約状況の確認、予約のキャンセルに関してTSUBAMEポータル利用の手引き 計算ノードの予約をご参照ください。

予約時間になりましたら、予約グループのアカウントでジョブの実行ができるようになります。 予約IDであるAR IDを指定したジョブ投入の例を以下に示します。

- qsubで予約ノードにジョブを投入する場合

```
[login]$ qsub -g [TSUBAMEグループ] -ar [AR ID] スクリプト名
```

・qrshで予約ノードにインタラクティブジョブを投入する場合

```
[login]$ qrsh -g [TSUBAMEグループ] -l [資源タイプ]=[個数] -l h_rt=[時間] -ar [AR ID]
```

予約実行で利用できる資源タイプはnode_f,node_h,node_q,node_oになります。その他の資源タイプは、予約では利用できません。

ジョブ投入後の ジョブの状態確認 は qstat コマンド、ジョブの削除はqdelコマンドを使用します。

また、スクリプトの書式は通常実行時のものと同じになります。

コマンドラインから予約状況及びAR IDを確認するためには `t4-user-info compute ar` を使用します。

```
[login]$ t4-user-info compute ar
ar_id   uid user_name         gid group_name                state     start_date          end_date        time_hour node_count      point return_point
-------------------------------------------------------------------------------------------------------------------------------------------------------
 1320  2005 A2901247         2015 tga-red000                 r   2023-01-29 12:00:00 2023-01-29 13:00:00         1          1      18000            0
 1321  2005 A2901247         2015 tga-red000                 r   2023-01-29 13:00:00 2023-01-29 14:00:00         1          1      18000            0
 1322  2005 A2901247         2015 tga-red000                 w   2023-01-29 14:00:00 2023-02-02 14:00:00        96          1    1728000      1728000
 1323  2005 A2901247         2015 tga-red000                 r   2023-01-29 14:00:00 2023-02-02 14:00:00        96          1    1728000      1728000
 1324  2005 A2901247         2015 tga-red000                 r   2023-01-29 15:00:00 2023-01-29 16:00:00         1         17     306000            0
 1341  2005 A2901247         2015 tga-red000                 w   2023-02-25 12:00:00 2023-02-25 13:00:00         1         18     162000       162000
 3112  2004 A2901239         2349 tgz-training               r   2023-04-24 12:00:00 2023-04-24 18:00:00         6         20     540000            0
 3113  2004 A2901239         2349 tgz-training               r   2023-04-25 12:00:00 2023-04-25 18:00:00         6         20     540000            0
 3116  2005 A2901247         2015 tga-red000                 r   2023-04-18 17:00:00 2023-04-25 16:00:00       167          1    3006000            0
 3122  2005 A2901247         2014 tga-blue000                r   2023-04-25 08:00:00 2023-05-02 08:00:00       168          5   15120000            0
 3123  2005 A2901247         2014 tga-blue000                r   2023-05-02 08:00:00 2023-05-09 08:00:00       168          5    3780000            0
 3301  2005 A2901247         2015 tga-red000                 r   2023-08-30 14:00:00 2023-08-31 18:00:00        28          1     504000            0
 3302  2005 A2901247         2009 tga-green000               r   2023-08-30 14:00:00 2023-08-31 18:00:00        28          1     504000            0
 3304  2005 A2901247         2014 tga-blue000                r   2023-09-03 10:00:00 2023-09-04 10:00:00        24          1     432000            0
 3470  2005 A2901247         2014 tga-blue000                w   2023-11-11 22:00:00 2023-11-11 23:00:00         1          1       4500         4500
 4148  2004 A2901239         2007 tga-hpe_group00            w   2024-04-12 17:00:00 2024-04-12 18:00:00         1          1       4500         4500
 4149  2005 A2901247         2015 tga-red000                 w   2024-04-12 17:00:00 2024-04-13 17:00:00        24          1     108000       108000
 4150  2004 A2901239         2007 tga-hpe_group00            w   2024-04-12 17:00:00 2024-04-12 18:00:00         1          1       4500         4500
-------------------------------------------------------------------------------------------------------------------------------------------------------
total :                                                                                                          818         97   28507500      3739500
```

コマンドラインから当月の予約の空き状況を確認するには、 `t4-user-info compute ars` を使用します。

5.2.6. 計算ノードへのSSHログイン

資源タイプ node_f でジョブを行ったノードには直接sshでログインできます。
確保したノードは以下の手順により、確認することができます。

```
[login]$ qstat -j 1463
==============================================================
job_number:                 1463
(途中省略)
exec_host_list        1:    r8n3:28, r8n4:28       ← 確保したノード  r8n3、r8n4
(以降省略)
```

計算ノードにsshした際はsshしたプロセスのGIDがtsubame-users(2000)となっている為、お試し実行の場合を除いて、ssh直後の状態では実行している自分のジョブのプロセスが見えず、gdbでアタッチもできません。
見えるようにするにはssh後にジョブを実行したグループ名で以下を実行して下さい。

`newgrp <グループ名>`
又は
`sg <グループ名>`

5.3. インタラクティブジョブ専用キュー

インタラクティブジョブ専用キューは、同じリソースをユーザ間で共有することで、混雑時にもノード確保に失敗しにくく、可視化や対話的なプログラムを素早く開始できるように用意された環境です。

インタラクティブジョブ専用キューへのジョブの投入方法は以下のとおりです。

```
[login]$ iqrsh -g [TSUBAMEグループ] -l h_rt=<時間>
```

Info

インタラクティブジョブ専用キューは、学内ユーザ(tgz-edu)とアクセスカードユーザに限り無償で実行可能です。
無償利用する場合、**-g [TSUBAMEグループ]** オプションを指定せずにジョブを投入してください。
-g オプションを指定した場合、対象グループに対して課金が発生しますのでご注意ください。

※インタラクティブジョブ専用キューではCPU/GPUのオーバーコミットが許可されていることにご注意下さい。
インタラクティブジョブ専用キューの制限値一覧はこちらをご確認下さい。

5.4. 計算ノード上のストレージの利用

5.4.1. ローカルスクラッチ領域

SSDをローカルスクラッチ領域として使用することができます。

ローカルスクラッチ領域を利用することで、ノード間を跨がず最も早いファイルアクセスが実現できます。
ただし、ジョブの先頭にてグループディスクやホームディレクトリからローカルスクラッチ領域へのファイル転送処理を行う必要があるため、転送するファイルが大量であったりアクセス頻度が低い場合、結果的に実行時間が遅くなる可能性もありますのでご注意ください。
ローカルスクラッチ領域上のファイルはジョブの終了時に消去されるため、必要なファイルはユーザーが明示的に Homeディレクトリまたはグループディスクに保存します。

各資源タイプで利用可能なローカルスクラッチ領域の容量は、こちらをご覧下さい。

ローカルスクラッチ領域は環境変数 T4TMPDIR を指定することにより利用可能です。

ローカルスクラッチ領域は各計算ノードの個別領域となり共有されていないため、ジョブスクリプト内からの入力ファイルと出力ファイルをローカルホストにステージングする必要があります。

下記の例では、使用する計算ノードが1ノードの場合に、ホームディレクトリからローカルスクラッチ領域にインプットデータセットをコピーし、結果をホームディレクトリに返します。 (複数ノードは対応していません)

```
#!/bin/sh

cp -rp $HOME/datasets ${T4TMPDIR}                     # 計算に必要な入力ファイルのコピー
./a.out ${T4TMPDIR}/datasets ${T4TMPDIR}/results  # 入力、出力を指定する計算プログラムの実行
cp -rp ${T4TMPDIR}/results $HOME/results              # 必要な結果ファイルのコピー
```

Tips

${T4TMPDIR} はジョブ終了時に削除されます。

5.4.2. /tmp領域

/tmpディレクトリは1ノードあたり約91GiB確保されていますが、該当ノードを利用する全てのジョブおよびシステムで共有利用します。
そのため、1つのジョブで91GiB全てを利用できる保証はありません。
環境変数TMPDIRは/tmp以下のジョブごとに固有のディレクトリになりますが、大容量スクラッチファイルの作成時などの際に実行プログラムのハングアップなどの問題が発生する懸念があります。

Warning

/tmp 直下は利用禁止です。原則ローカルスクラッチ領域を利用し、必要に応じて環境変数 TMPDIR を利用してください。

6. 商用アプリケーション

Info

本ページのコマンドライン例では、以下の表記を使用します。
[login]$: ログインノード
[rNnN]$: 計算ノード
[login/rNnN]$: ログインノードまたは計算ノード
[yourPC]$: ログインノードへの接続元環境

ライセンス契約上、商用アプリケーションを利用できる利用者は限られます。 東京科学大に所属する「1.学生証・職員証」以外の利用者は以下の商用アプリケーションのみ利用できます。

- Gaussian/Gauss View
- AMBER　学術機関に所属する利用者に限る
- Intel oneAPI Compiler
- NVIDIA HPC SDK Compiler
- Linaro forge(旧:Arm Forge)

Info

一部の商用アプリケーションの利用には別途アプリケーション利用料が必要になります。詳細は利用料の概略のアプリケーション **(TSUBAME4.0**で一部有償化)をご覧下さい。

商用アプリケーションの一覧表を以下に示します。

ソフトウェア名	概要
ANSYS	解析ソフトウェア
ANSYS/Fluent	解析ソフトウェア
ANSYS/LS-DYNA	解析ソフトウェア
ABAQUS	解析ソフトウェア
ABACUS CAE	解析ソフトウェア
Gaussian	量子化学計算プログラム
GaussView	量子化学計算プログラム プリポストツール
AMBER	分子動力学計算プログラム
Materials Studio	化学シミュレーションソフトウェア
Discovery Studio	化学シミュレーションソフトウェア
Mathematica	数式処理ソフトウェア
COMSOL	解析ソフトウェア
Schrodinger	化学シミュレーションソフトウェア
MATLAB	数値計算ソフトウェア
VASP	量子分子動力学計算プログラム
Linaro forge(旧:Arm Forge)	デバッガ
Intel oneAPI Compiler	コンパイラ
NVIDIA HPC SDK Compiler	コンパイラ

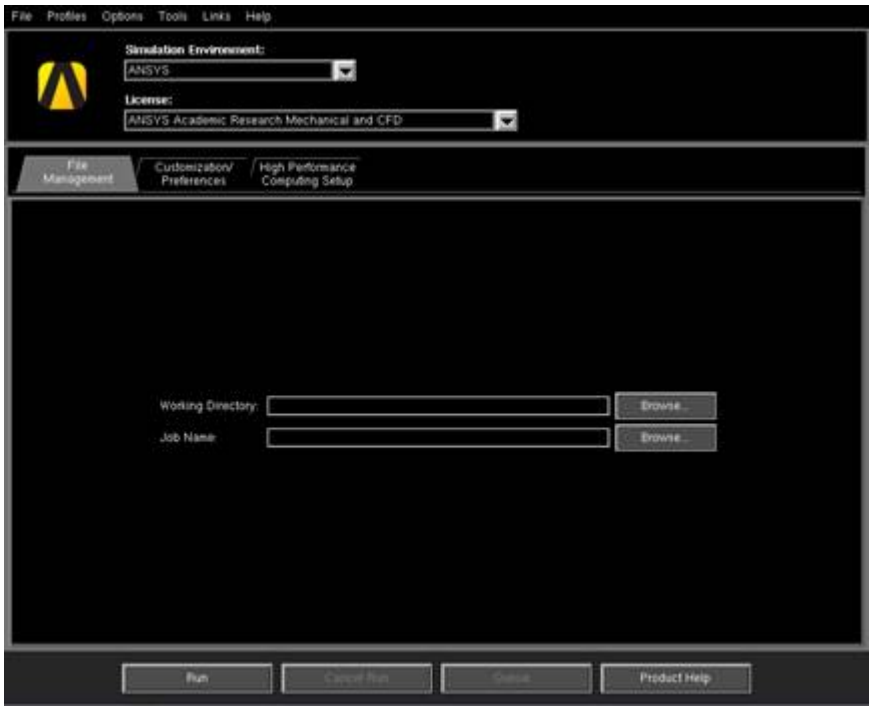
Info

本ページで使用しているmoduleコマンドについては利用環境の切換え方法を参照してください。

6.1. ANSYS

GUIでの利用手順を以下に示します。

```
[rNnN]$ module load ansys
[rNnN]$ launcher
```



CLIでの利用手順を以下に示します。

```
[rNnN]$ module load ansys
[rNnN]$ mapdl
```

mapdlコマンドの代わりに以下のコマンドも使用できます。

　ANSYS24.1の場合。バージョンによって異なります。

```
[rNnN]$ ansys241
```

`exit` と入力すると終了します。

入力ファイルを指定すると非対話的に実行されます。

実行例1

```
[rNnN]$ mapdl [options] < inputfile > outputfile
```

実行例2

```
[rNnN]$ mapdl [options] -i inputfile -o outputfile
```

バッチキューシステムを使用する場合は、シェルスクリプトを作成しCLIで以下のように実行します。

sample.shを使用する場合

```
[login]$ qsub sample.sh
```

スクリプト例　MPI並列処理

```
#!/bin/bash
#$ -cwd
#$ -V
#$ -l node_f=2
#$ -l h_rt=0:10:0

module load ansys

mapdl -b -dis -np 56 < inputfile > outputfile
```

スクリプト例　GPU使用

```
#!/bin/bash
#$ -cwd
#$ -V
#$ -l node_f=1
#$ -l h_rt=0:10:0

module load ansys

mapdl -b -dis -np 28 -acc nvidia -na 4 < inputfile > outputfile
```

ANSYSのライセンス利用状況を以下のコマンドで確認できます。

```
[login]$ lmutil lmstat -S ansyslmd -c *****@kvm5.ini.t4.gsic.titech.ac.jp:*****@kvm6.ini.t4.gsic.titech.ac.jp:*****@ldap2.ini.t4.gsic.titech.ac.jp
```

Info

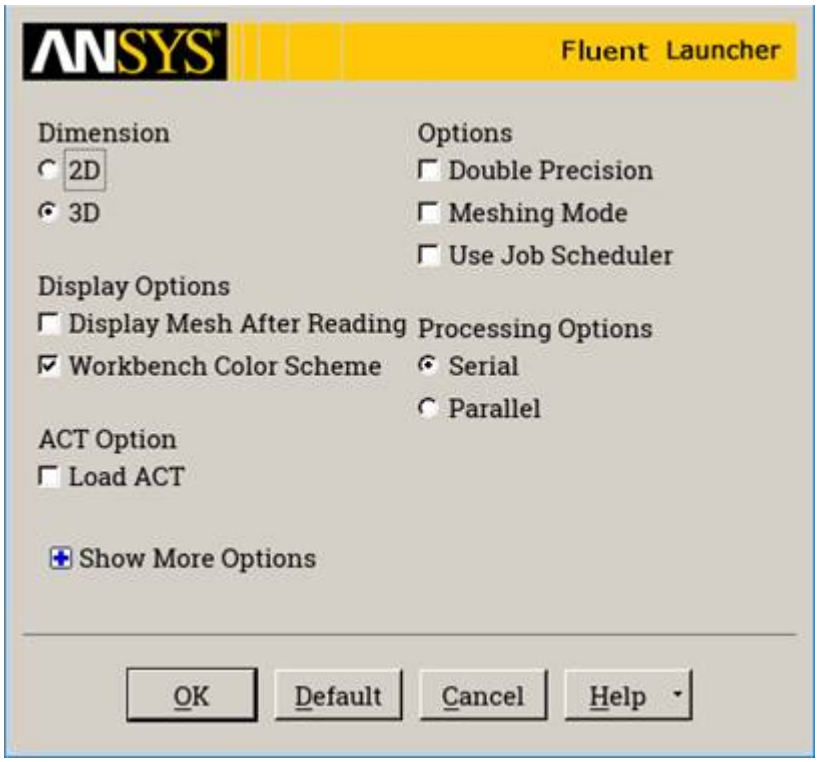
ポート番号はアプリケーション有効化(TSUBAME4)で購入後に参照可能です。
/apps/t4/rhel9/isv/ansys_inc/t4-license
ポート番号は毎年4月中旬頃変更されます。

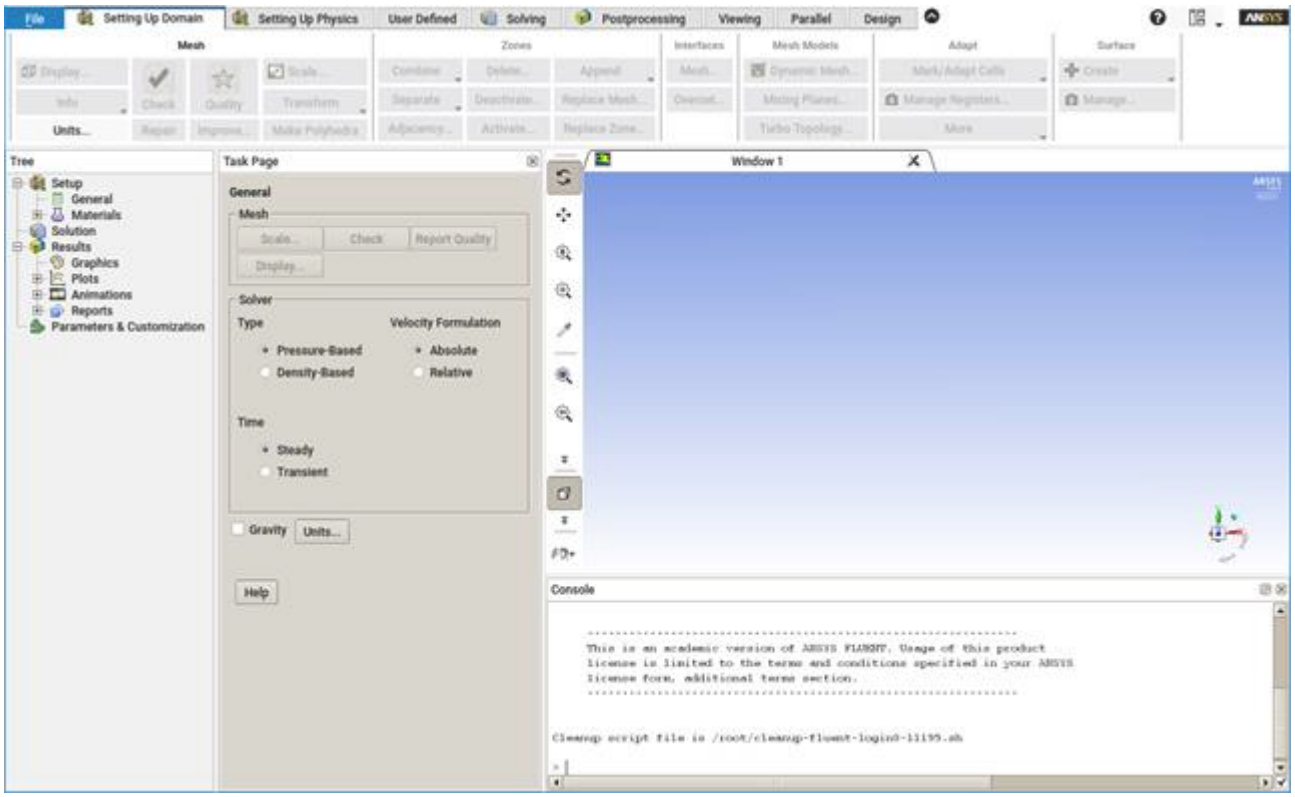
6.2. ANSYS/Fluent

ANSYS/Fluentは熱流体解析アプリケーションです。利用手順を以下に示します。

GUIでの起動手順を以下に示します。

```
[rNnN]$ module load ansys
[rNnN]$ fluent
```





CLIでの起動手順を以下に示します。

```
[rNnN]$ module load ansys
[rNnN]$ fluent -g
```

exitと入力すると終了します。

journalファイルを使用してインタラクティブに実行する場合は以下のようにコマンドを実行します。

journalファイル名がfluentbench.jou、3Dの場合

```
[rNnN]$ fluent 3d -g -i fluentbench.jou
```

バッチキューシステムを使用する場合は、シェルスクリプトを作成しCLIで以下のように実行します。

sample.shを利用する場合

```
[login]$ qsub sample.sh
```

SMP並列、MPI並列、及びGPUを使用する場合のバッチジョブスクリプト例を以下に示します。

バッチスクリプト例　SMP

```
#!/bin/bash
#$ -cwd
#$ -l node_f=1
#$ -l h_rt=:10:

module load ansys

JOURNAL=rad_a_1.jou
OUTPUT=rad_a_1.out
VERSION=3d

fluent -mpi=intel -g ${VERSION} -scheduler_pe=${PE_HOSTFILE} -i ${JOURNAL} > ${OUTPUT} 2>&1
```

バッチスクリプト例　MPI、CPU

```
#!/bin/bash
#$ -cwd
#$ -l node_f=1
#$ -l h_rt=:10:

module load ansys

JOURNAL=rad_a_1.jou
OUTPUT=rad_a_1.out
VERSION=3d
NCPUS=192

fluent -mpi=intel -g ${VERSION} -t${NCPUS} -scheduler_pe=${PE_HOSTFILE} -i ${JOURNAL} > ${OUTPUT} 2>&1
```

バッチスクリプト例　MPI、GPU

```
#!/bin/bash
#$ -cwd
#$ -l node_f=1
#$ -l h_rt=:10:

module load ansys

JOURNAL=rad_a_1.jou
OUTPUT=rad_a_1.out
VERSION=3d
GPGPU=4
NCPUS=192

fluent -mpi=intel -g ${VERSION} -t${NCPUS} -gpgpu=${GPGPU} -scheduler_pe=${PE_HOSTFILE} -i ${JOURNAL} > ${OUTPUT} 2>&1
```

Info

Fluentでは資源をまたぐ設定ができないため、 `#$ -l {資源名}=1` (例えば `node_f` では `#$ -l node_f=1`)としてください。
(TSUBAME3で指定していた -ncheck オプションは、v24では廃止されました。)

ANSYS/Fluentのライセンス利用状況を以下のコマンドで確認できます。

```
[login]$ lmutil lmstat -S ansyslmd -c *****@kvm5.ini.t4.gsic.titech.ac.jp:*****@kvm6.ini.t4.gsic.titech.ac.jp:*****@ldap2.ini.t4.gsic.titech.ac.jp
```

Info

ポート番号はアプリケーション有効化(TSUBAME4)で購入後に参照可能です。
/apps/t4/rhel9/isv/ansys_inc/t4-license
ポート番号は毎年4月中旬頃変更されます。

6.3. ANSYS/LS-DYNA

6.3.1. ANSYS/LS-DYNAの概要

LS-DYNAは、陽解法により構造物の大変形挙動を時刻履歴で解析するプログラムで、衝突　衝撃解析、落下解析、塑性加工解析、貫通　亀裂　破壊解析などに威力を発揮し、これらの分野では世界有数の導入実績を誇る信頼性の高いプログラムです。

6.3.2. ANSYS/LS-DYNAの使用方法

ANSYS/LS-DYNAはバッチジョブで利用します。バッチスクリプトの例を以下に示します。

使用したいバージョンに適宜読み替えてご実行ください。

スクリプト例　MPP単精度版

```
#!/bin/bash
#$ -cwd
#$ -V
#$ -l node_h=1
#$ -l h_rt=5:00:0


module load ansys intel-mpi

export dynadir=/apps/t4/rhel9/isv/ansys_inc/v241/ansys/bin/linx64
export exe=$dynadir/lsdyna_sp_mpp.e
export dbo=$dynadir/lsl2a_sp.e

export NCPUS=4
export INPUT=$base_dir/sample/airbag_deploy.k

mpiexec -np ${NCPUS} ${exe} i=${INPUT}

${dbo} binout*
```

スクリプト例　MPP倍精度版

```
#!/bin/bash
#$ -cwd
#$ -V
#$ -l node_h=1
#$ -l h_rt=5:00:0

module load ansys intel-mpi

export dynadir=/apps/t4/rhel9/isv/ansys_inc/v241/ansys/bin/linx64
export exe=$dynadir/lsdyna_dp_mpp.e
export dbo=$dynadir/lsl2a_dp.e

export NCPUS=4
export INPUT=$base_dir/sample/airbag_deploy.k

mpiexec -np ${NCPUS} ${exe} i=${INPUT}

${dbo} binout*
```

スクリプトは、利用者の環境に合わせて変更してください。 上記スクリプト例では、インプットファイルはシェルスクリプト内でINPUT=inputfileとして指定しています。

6.4. ABAQUS

インタラクティブでの利用手順を以下に示します。

```
[rNnN]$ module load abaqus
[rNnN]$ abaqus job=inputfile [options]
```

バッチキューシステムを使用する場合は、シェルスクリプトを作成しCLIで以下のように実行します。

sample.shを利用する場合

```
[login]$ qsub sample.sh
```

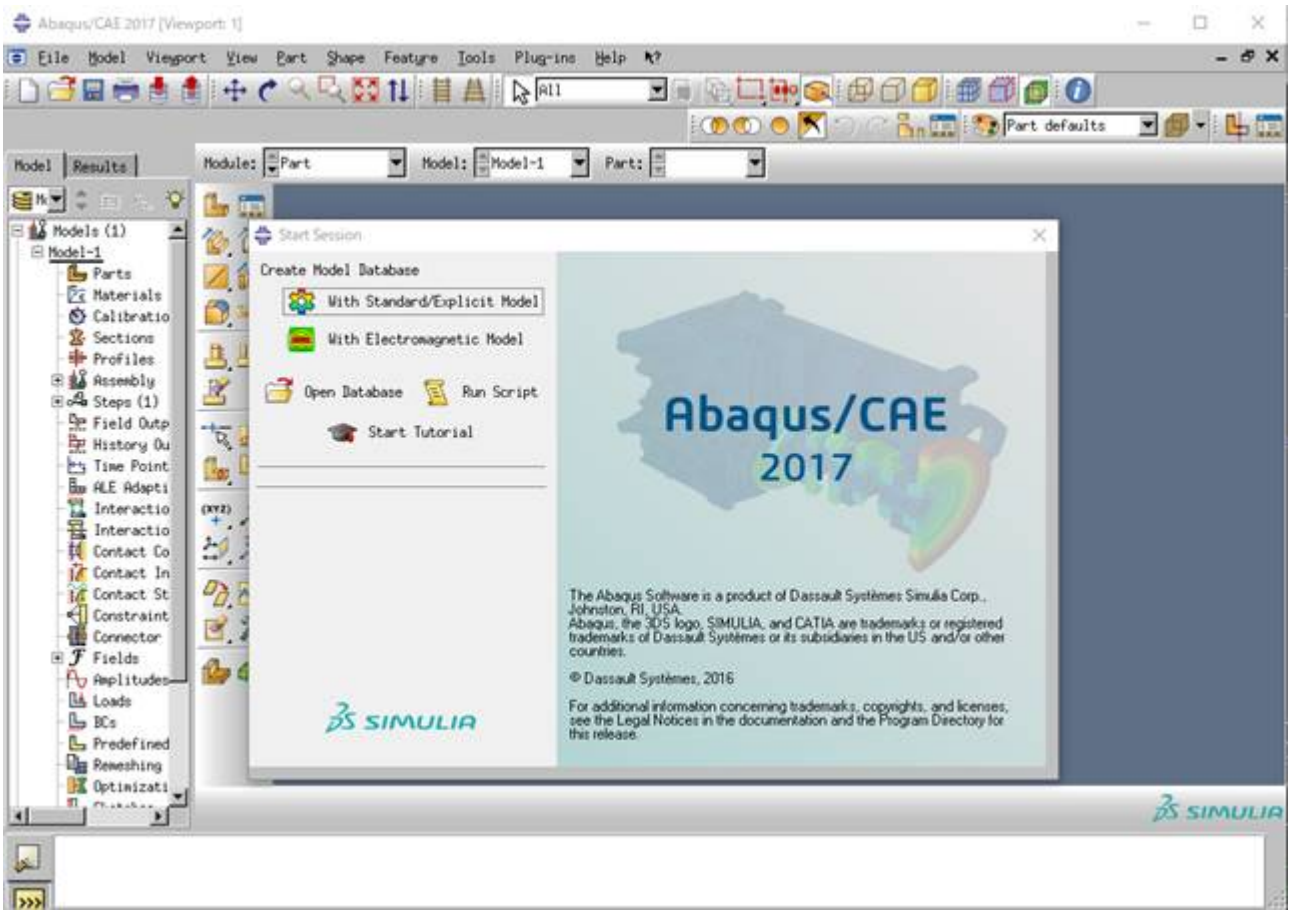
スクリプト例　MPI並列処理

```
#!/bin/bash
#$ -cwd
#$ -V
#$ -l node_o=1
#$ -l h_rt=0:10:0


module load abaqus

# ABAQUS settings.
INPUT=s2a
ABAQUS_VER=2024
ABAQUS_CMD=abq${ABAQUS_VER}
SCRATCH=${T4TMPDIR}
NCPUS=4

${ABAQUS_CMD} interactive \
job=${INPUT} \
cpus=${NCPUS} \
scratch=${SCRATCH} \
mp_mode=mpi > ${INPUT}.`date '+%Y%m%d%H%M%S'`log 2>&1
```

6.5. ABAQUS CAE

ABAQUS CAEの利用手順を以下に示します。

```
[rNnN]$ module load abaqus
[rNnN]$ abaqus cae
```



メニューバーの File > Exit をクリックすると終了します。

6.6. Gaussian

インタラクティブな利用手順を以下に示します。

```
[rNnN]$ module load gaussian/revision
[rNnN]$ g16 inputfile
```

revisionには使用するリビジョンを指定してください。Gaussian16 Rev.C02の場合は以下の通りです。

```
[rNnN]$ module load gaussian/16C2_cpu
[rNnN]$ g16 inputfile
```

バッチキューシステムを使用する場合は、シェルスクリプトを作成しCLIで以下のように実行します。

sample.shを使用する場合

```
[login]$ qsub sample.sh
```

スクリプト例　ノード内並列処理

Glycineの構造最適化および振動解析(IR+ラマン強度)を計算する場合のサンプルスクリプトです。

下記のglycine.sh、glycine.gjfを同一ディレクトリ上に配置し、下記コマンドを実行することで計算ができます。計算後にglycine.log、glycine.chkが生成されます。

解析結果の確認についてはGaussViewにてご説明します。

```
[login]$ qsub glycine.sh
```

glycine.sh

```
#!/bin/bash
#$ -cwd
#$ -l node_f=1
#$ -l h_rt=0:10:0
#$ -V


module load gaussian

g16 glycine.gjf
```

glycine.gjf

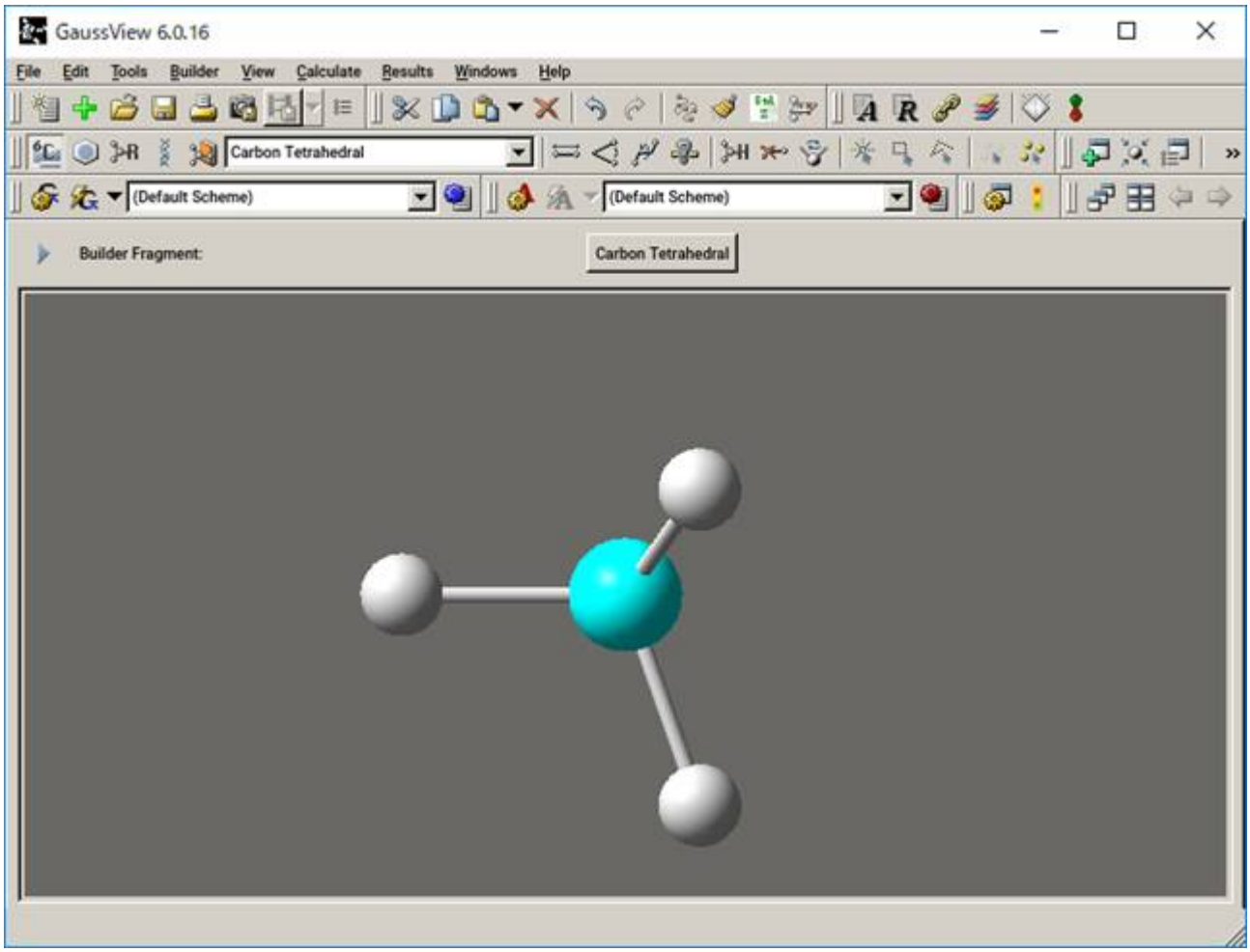
```
chk=glycine.chk
cpu=0-191      ←環境変数GAUSS_CDEF/GAUSS_GDEFを自動設定するモジュールを読み込んだ場合は不要
mem=700GB

P opt=(calcfc,tight,rfo) freq=(raman)

glycine Test Job

 2
N                 0   -2.15739574   -1.69517043   -0.01896033 H
H                 0   -1.15783574   -1.72483643   -0.01896033 H
C                 0   -2.84434974   -0.41935843   -0.01896033 H
C                 0   -1.83982674    0.72406557   -0.01896033 H
H                 0   -3.46918274   -0.34255543   -0.90878333 H
H                 0   -3.46918274   -0.34255543    0.87086267 H
O                 0   -0.63259574    0.49377357   -0.01896033 H
O                 0   -2.22368674    1.89158057   -0.01896033 H
H                 0   -2.68286796   -2.54598119   -0.01896033 H

1 2 1.0 3 1.0 9 1.0
2
3 4 1.0 5 1.0 6 1.0
4 7 1.5 8 1.5
5
6
7
8
9
```

6.7. GaussView

GaussViewはGaussianの結果を可視化するアプリケーションです。

GaussViewの利用手順を以下に示します。

```
[rNnN]$ module load gaussian gaussview
[rNnN]$ gview.exe
```



メニューバーから File > Exit をクリックすると終了します。

解析例　glycine.log

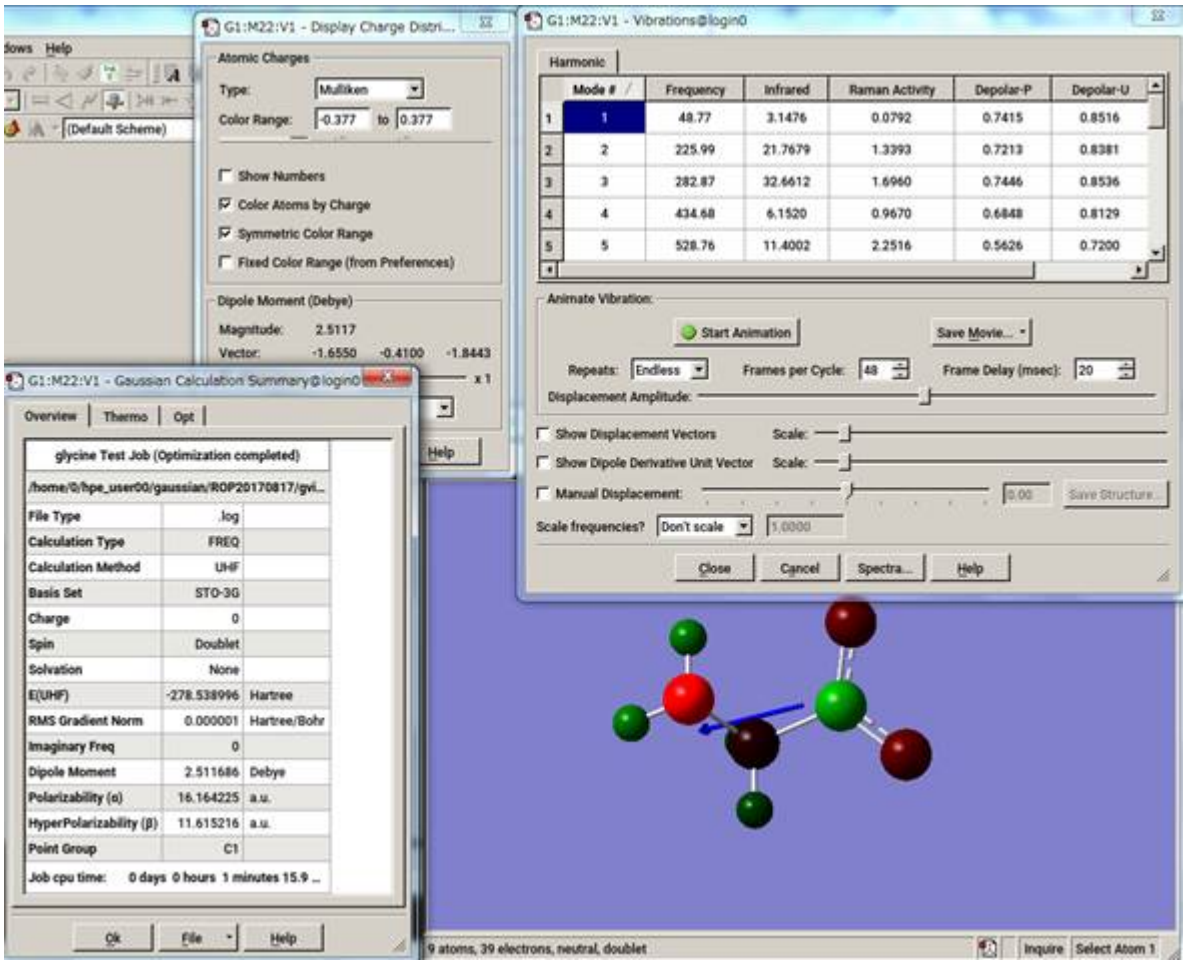
Gaussianの項にてサンプルとして例示しているスクリプトを実行した結果ファイルの解析を例にご説明します。

```
[rNnN]$ module load gaussian gaussview
[rNnN]$ gview.exe glycine.log
```

Resultから解析結果の確認が可能です。

Result>Summaryにて計算の概要、Result>ChageDistribution…で電荷情報、Vibration…から振動解析の結果を確認できます。

サンプルでは振動解析を行っているので、VibrationダイアログのStartAnimationから振動の様子を確認できます。



6.8. AMBER

AMBERは本来タンパク質・核酸の分子動力学計算のために開発されたプログラムですが、最近では糖用のパラメータも開発され、化学・生物系の研究のために益々有用なツールとなってきました。ご自分の研究で利用する場合は、マニュアルや関 連する論文等の使用例をよく調べて、AMBERが採用しているモデルや理論の限界、応用範囲等を把握しておくことが必要です。現在、AMBERはソースコードを無制限にコピーすることはできませんが、東京科学大内部で利用することは可能なので、これを基にさらに発展した手法を取り込むことも可能です。

下記について、使用したいバージョンに適宜読み替えてご実行ください。

インタラクティブでの逐次処理の場合の利用手順を以下に示します。

```
[rNnN]$ module load amber
[rNnN]$ sander [-O|A] -i mdin -o mdout -p prmtop -c inpcrd -r restrt
```

インタラクティブでの並列処理(sander.MPI)の場合の利用手順を以下に示します。

```
[rNnN]$ module load amber
[rNnN]$ mpirun -np -[並列数] -x LD_LIBRARY_PATH sander.MPI [-O|A] -i mdin -o mdout -p prmtop -c inpcrd -r restrt
```

インタラクティブでのGPU逐次処理(pmemd.cuda)の場合の利用手順を以下に示します。

```
[rNnN]$ module load amber
[rNnN]$ pmemd.cuda [-O] -i mdin -o mdout -p prmtop -c inpcrd -r restrt
```

インタラクティブでのGPU並列処理(pmemd.cuda.MPI)の場合の利用手順を以下に示します。

```
[rNnN]$ module load amber
[rNnN]$ mpirun -np -[並列数] -x LD_LIBRARY_PATH pmemd.cuda.MPI [-O] -i mdin -o mdout -p prmtop -c inpcrd -r restrt
```

バッチキューシステムの場合の利用手順を以下に示します。

parallel.shを利用する場合

```
[login]$ qsub parallel.sh
```

スクリプト例　CPU並列処理

```
#!/bin/bash
#$ -cwd
#$ -l node_f=2
#$ -l h_rt=0:10:00
#$ -V
export NSLOTS=56

in=./mdin
out=./mdout_para
inpcrd=./inpcrd
top=./top

cat <<eof > $in
 Relaxtion of trip cage using
&cntrl
  imin=1,maxcyc=5000,irest=0, ntx=1,
  nstlim=10, dt=0.001,
  ntc=1, ntf=1, ioutfm=1
  ntt=9, tautp=0.5,
  tempi=298.0, temp0=298.0,
  ntpr=1, ntwx=20,
  ntb=0, igb=8,
  nkija=3, gamma_ln=0.01,
  cut=999.0,rgbmax=999.0,
  idistr=0
 /
eof


module load amber

mpirun -np $NSLOTS -x LD_LIBRARY_PATH \
sander.MPI -O -i $in -c $inpcrd -p $top -o $out < /dev/null

/bin/rm -f $in restrt
```

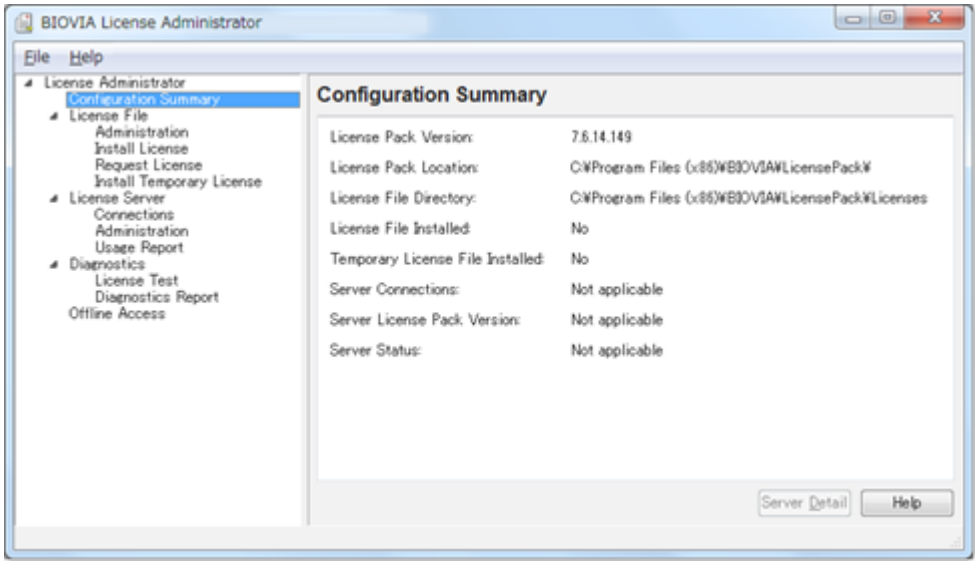
スクリプト例　GPU並列処理

```
#!/bin/bash
#$ -cwd
#$ -l node_f=2
#$ -l h_rt=0:10:0
#$ -V

export NSLOTS=56

in=./mdin
out=./mdout
inpcrd=./inpcrd
top=./top

cat <<eof > $in
FIX (active) full dynamics ( constraint dynamics: constant volume)
&cntrl
   ntx = 7,       irest = 1,
   ntpr = 100,     ntwx = 0,     ntwr = 0,
   ntf = 2,       ntc = 2,       tol = 0.000001,
   cut = 8.0,
   nstlim = 500,  dt = 0.00150,
   nscm = 250,
   ntt = 0,
   lastist = 4000000,
   lastrst = 6000000,
 /
eof


module load amber

mpirun -np $NSLOTS -x LD_LIBRARY_PATH \
pmemd.cuda.MPI -O -i $in -c $inpcrd -p $top -o $out < /dev/null

/bin/rm -f $in restrt
```

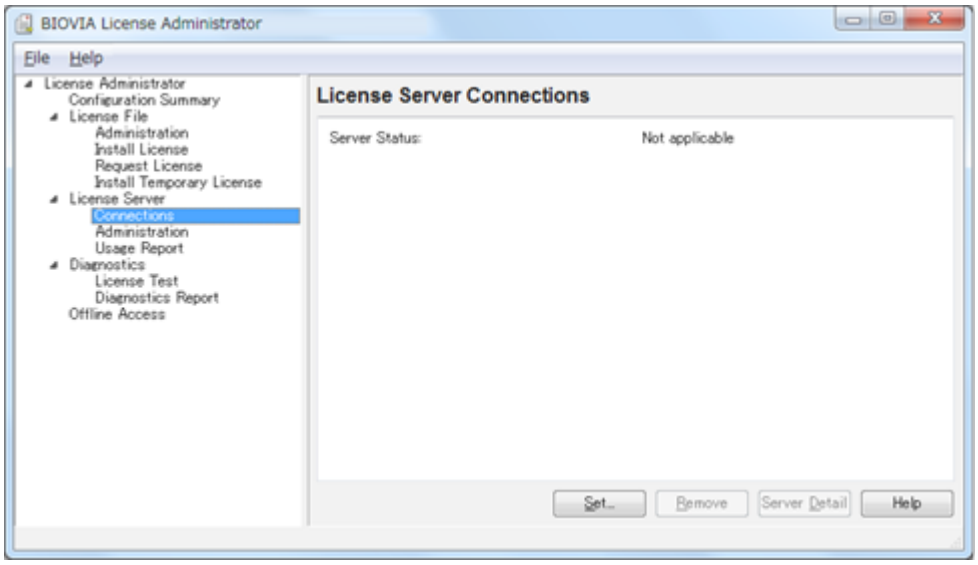
6.9. Materials Studio

6.9.1. ライセンス接続設定方法

スタートメニューから すべてのプログラム > BIOVIA > Licensing > License Administrator X.X.X を管理者として実行します。



[Connections] を開き、[Set] をクリックして Set License Server ダイアログを開きます。



Redundant servers にチェックを入れ、ホスト名を下表のように入力し、[OK] をクリックします。

Host name	kvm5.ini.t4.gsic.titech.ac.jp
Host name	kvm6.ini.t4.gsic.titech.ac.jp
Host name	ldap2.ini.t4.gsic.titech.ac.jp

Info

ポート番号は研究室向けアプリケーション配布で購入後に参照可能です。インストールマニュアルに記載されています。
ポート番号は毎年4月中旬頃変更されます。

Server Status が Connected と表示されれば設定完了です。

※Materials Studioを利用するためには、2ホスト以上のライセンスサーバーへの接続が確立している必要があります。

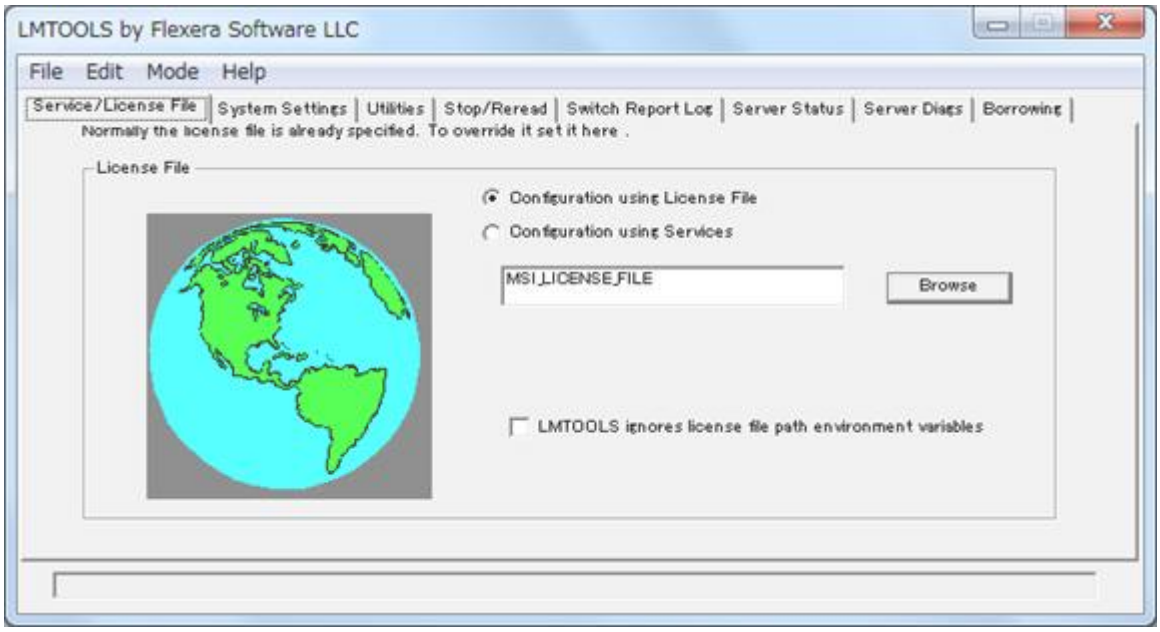
6.9.2. ライセンス利用状況の確認方法

6.9.2.1. Windowsでの確認方法

スタートメニューから　すべてのプログラム > BIOVIA > Licensing > License Administrator 7.6.14 > Utilities (FLEXlm LMTOOLs)　を実行します。

[Service/License File] タブを開き、 [Configulation using License File] を選択します。

MSI_LICENSE_FILE と表示されていることを確認します。



[Server Status] タブを開き、[Perform Status Enqurity] をクリックすると、ライセンスの利用状況が表示されます。

特定のライセンスのみを表示したい場合は、[Individual Feature] に表示したいライセンス名を入力して　[Perform Status Enqurity] を実行します。

6.9.2.2. ログインノード上での確認方法

以下のコマンドを実行すると、利用状況が表示されます。

```
[login]$ lmutil lmstat -S msi -c *****@kvm5.ini.t4.gsic.titech.ac.jp,*****@kvm6.ini.t4.gsic.titech.ac.jp,*****@ldap2.ini.t4.gsic.titech.ac.jp
```

Info

ポート番号はアプリケーション有効化(TSUBAME4)で購入後に参照可能です。
/apps/t4/rhel9/isv/materialsstudio/t4-license
ポート番号は毎年4月中旬頃変更されます。

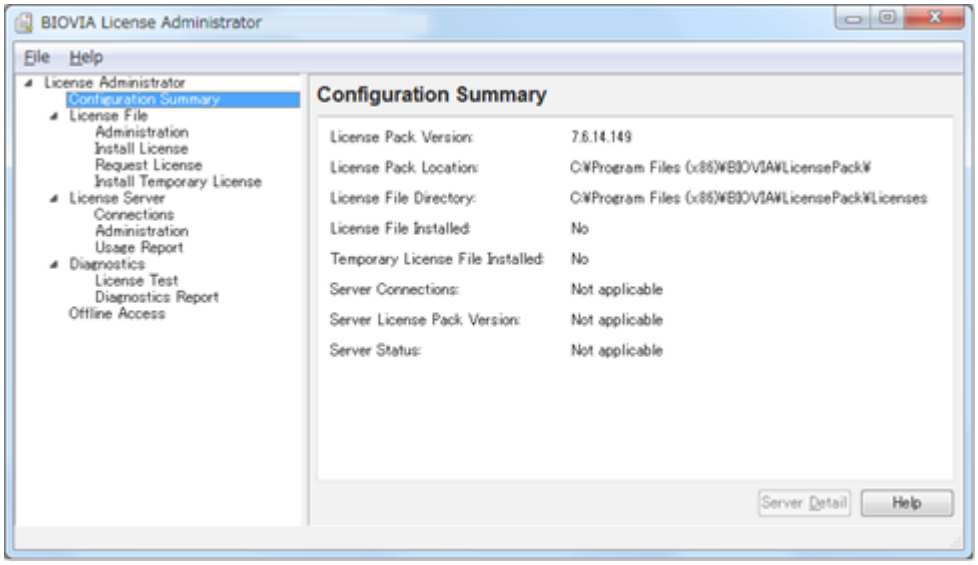
6.9.3. Materials Studioの起動方法

Materials Studioがインストールされたwindows環境においてスタートメニューを表示し、BIOVIA > Materials Studio 2024 をクリックして起動します。

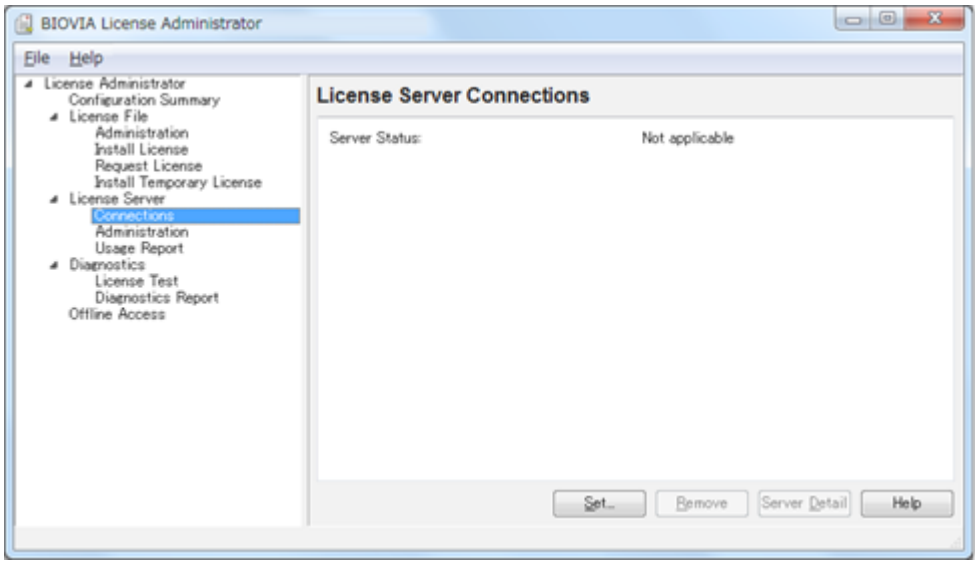
6.10. Discovery Studio

6.10.1. ライセンス接続設定方法

スタートメニューから　すべてのプログラム > BIOVIA > Licensing > License Administrator X.X.X　を管理者として実行します。



[Connections] を開き、[Set] をクリックして Set License Server ダイアログを開きます。



Redundant servers にチェックを入れ、ホスト名とポート番号を下図のように入力し、[OK] をクリックします。

Host name	kvm5.ini.t4.gsic.titech.ac.jp
Host name	kvm6.ini.t4.gsic.titech.ac.jp
Host name	ldap2.ini.t4.gsic.titech.ac.jp

Info

ポート番号は研究室向けアプリケーション配布で購入後に参照可能です。インストールマニュアルに記載されています。
ポート番号は毎年4月中旬頃変更されます。

Server Status が Connected と表示されれば設定完了です。

※Discovery Studioを利用するためには、2ホスト以上のライセンスサーバーへの接続が確立している必要があります。

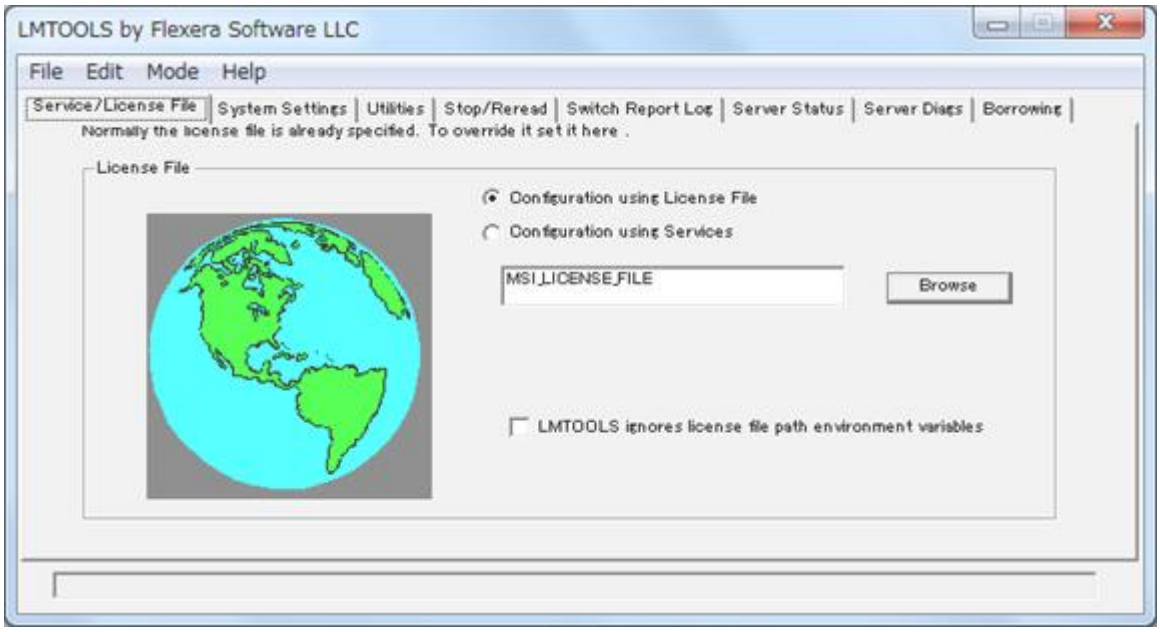
6.10.2. ライセンス利用状況の確認方法

6.10.2.1. Windowsでの確認方法

スタートメニューから　すべてのプログラム > BIOVIA > Licensing > License Administrator X.X.X > Utilities (FLEXlm LMTOOLs)　を実行します。

[Service/License File] タブを開き、 [Configulation using License File] を選択します。

MSI_LICENSE_FILE と表示されていることを確認します。



[Server Status] タブを開き、[Perform Status Enqurity] をクリックすると、ライセンスの利用状況が表示されます。

特定のライセンスのみを表示したい場合は、[Individual Feature] に表示したいライセンス名を入力して　[Perform Status Enqurity] を実行します。

6.10.3. Discovery Studioの起動方法

Discovery Studioがインストールされたwindows環境においてスタートメニューを表示し、BIOVIA > Discovery Studio 2024 64-bit Client をクリックして起動します。

6.11. Mathematica

Info

Mathematica バージョン14.1以降、アプリケーションの起動コマンド名が変更になりました。
利用するバージョンに合わせて、手順を参照してください。
デフォルトで有効になるバージョンおよび利用可能なバージョン一覧を確認する場合、以下のコマンドを実行します。

```
[login/rNnN]$ module avail mathematica
```

出力されたバージョン一覧のうち、下線が引かれたバージョンがデフォルトで有効になります。
出力例
mathematica/14.0 mathematica/14.1

6.11.1. Mathematica 14.0 利用時

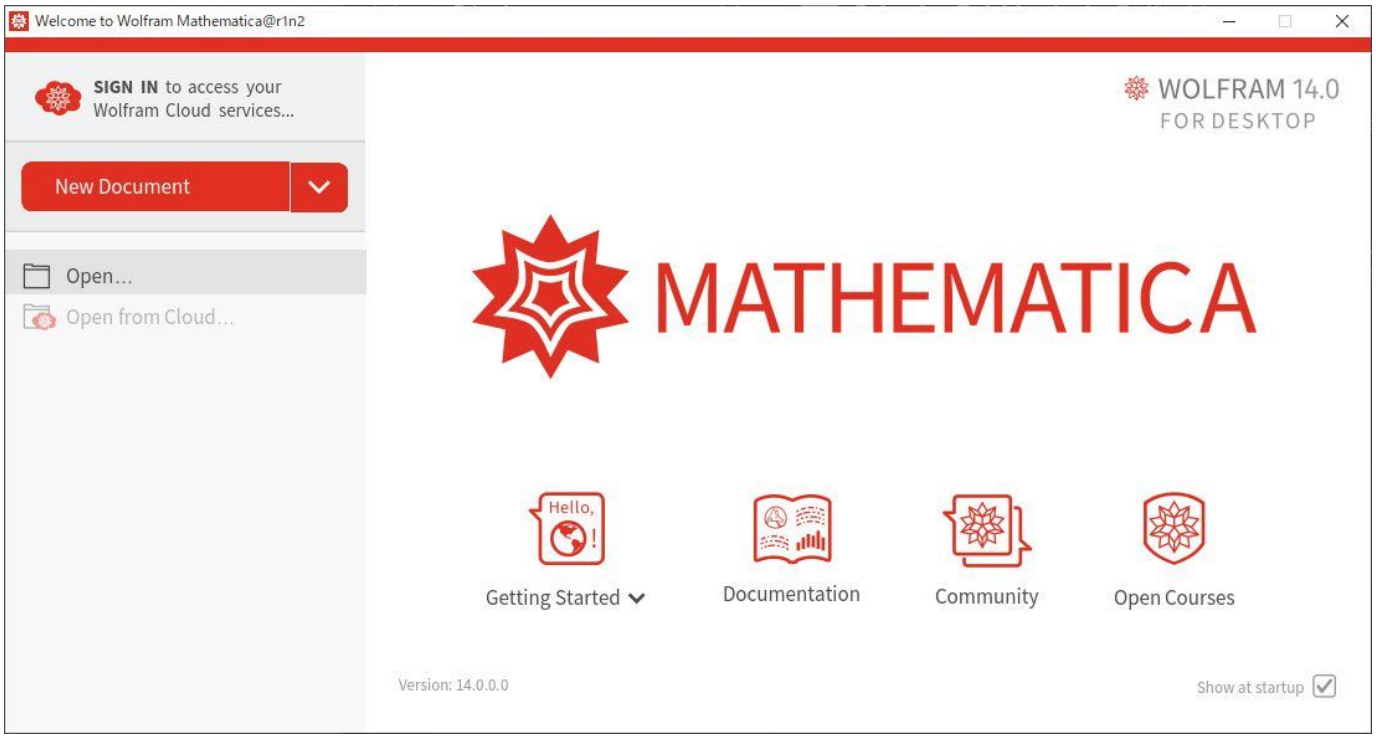
CLIでの起動手順を示します。

```
[rNnN]$ module load mathematica/14.0
[rNnN]$ math
Mathematica 14.0.0 Kernel for Linux x86 (64-bit)
Copyright 1988-2023 Wolfram Research, Inc.

In[1]:=
```

Quitと入力すると終了します。

GUIでの起動手順を以下に示します。

```
[rNnN]$ module load mathematica/14.0
[rNnN]$ Mathematica
```



6.11.2. Mathematica 14.1以降 利用時

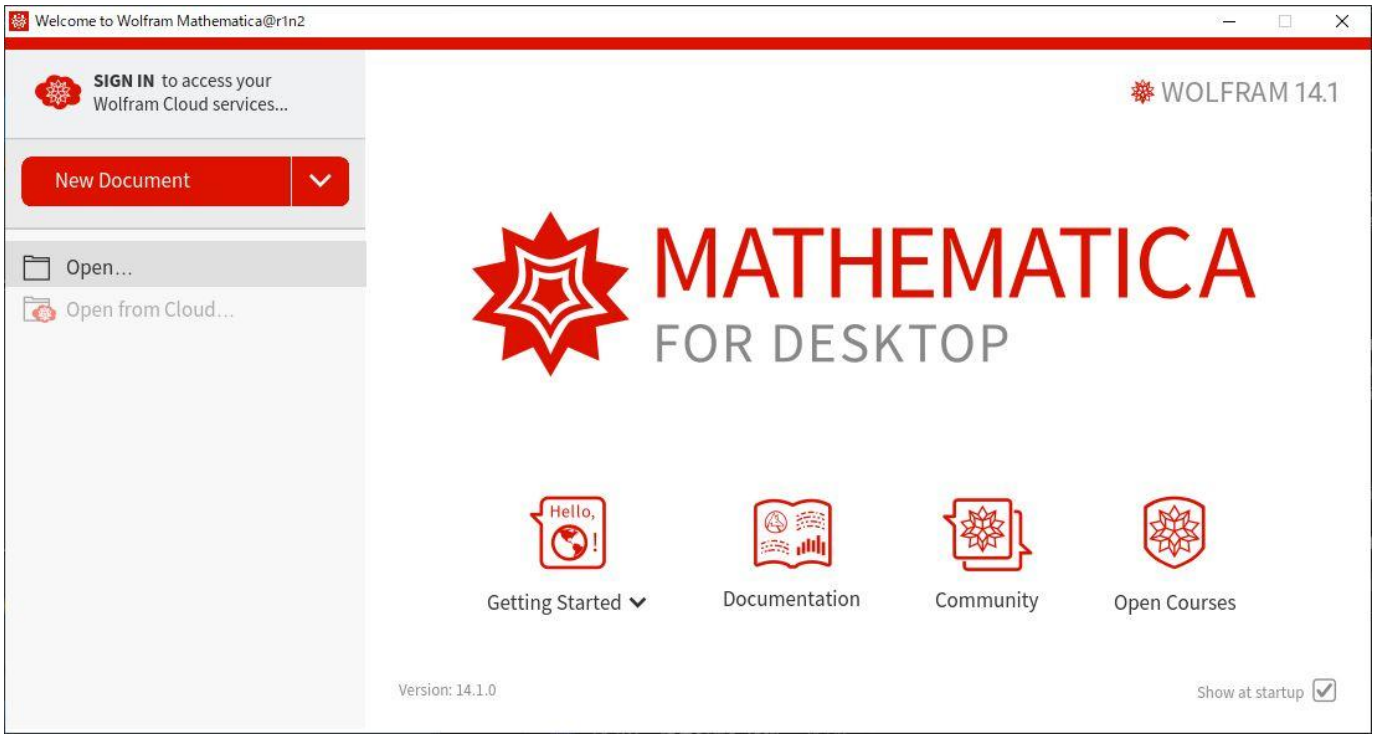
CLIでの起動手順を示します。(14.1 使用例)

```
[rNnN]$ module load mathematica/14.1
[rNnN]$ math
Wolfram 14.1.0 Kernel for Linux x86 (64-bit)
Copyright 1988-2024 Wolfram Research, Inc.


In[1]:=
```

Quitと入力すると終了します。

GUIでの起動手順を以下に示します。(14.1 使用例)

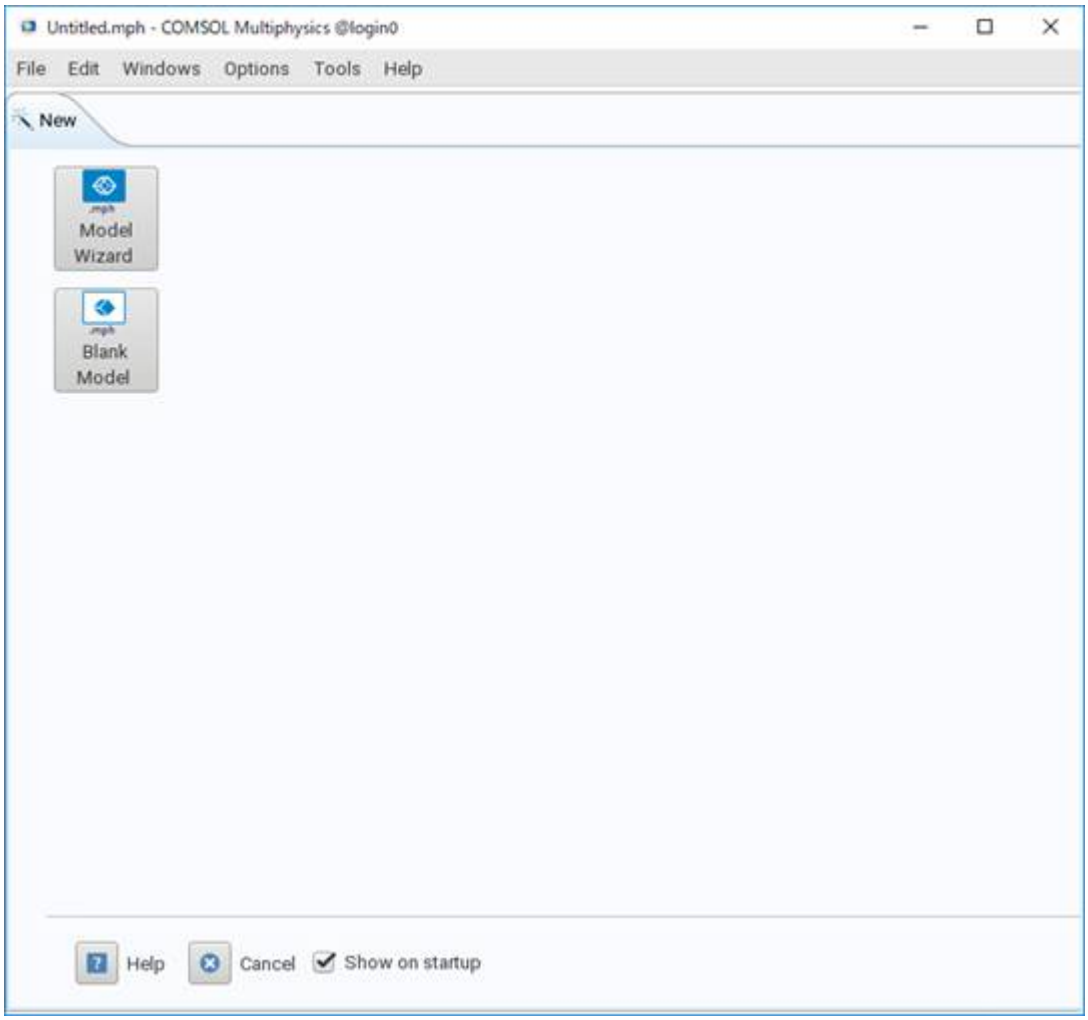
```
[rNnN]$ module load mathematica/14.1
[rNnN]$ WolframNB
```



6.12. COMSOL

COMSOLの利用手順を以下に示します。

```
[rNnN]$ module load comsol
[rNnN]$ comsol
```



メニューバーの File > Exit をクリックすると終了します。

COMSOLのライセンス利用状況を以下のコマンドで確認できます。

```
[login]$ lmutil lmstat -S LMCOMSOL -c *****@kvm5.ini.t4.gsic.titech.ac.jp:*****@kvm6.ini.t4.gsic.titech.ac.jp:*****@ldap2.ini.t4.gsic.titech.ac.jp
```

Info

ポート番号はアプリケーション有効化(TSUBAME4)で購入後に参照可能です。
/apps/t4/rhel9/isv/comsol/t4-license
ポート番号は毎年4月中旬頃変更されます。

6.13. Schrodinger

Schrodingerの利用手順を以下に示します。

LigprepのCLI実行例

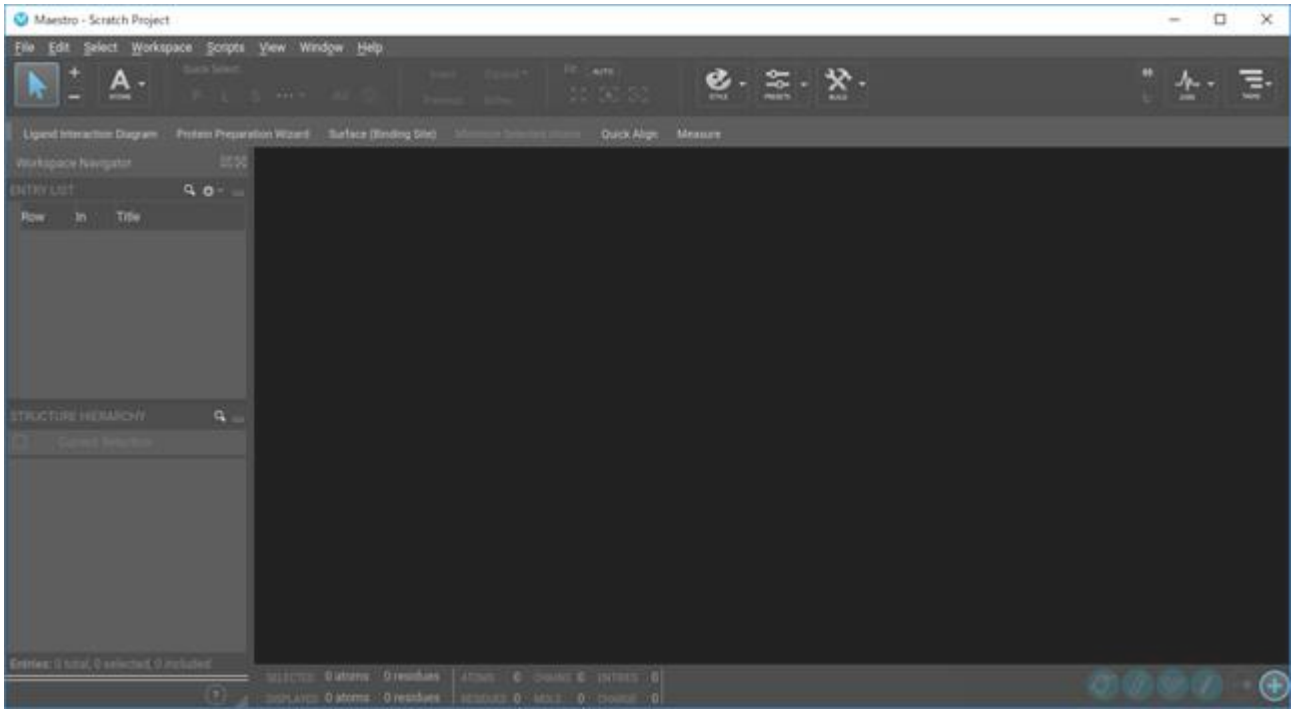
```
[rNnN]$ module load schrodinger
```

SMILES形式の入力ファイルを使用し、MAE形式で出力する場合

```
[rNnN]$ ligprep -ismiinputfile -omaeoutputfile
```

GUIで利用する場合は、Maestroを起動します。

```
[rNnN]$ module load schrodinger
[rNnN]$ maestro
```



メニューバーのFile > Exit をクリックすると終了します。

Schrodingerのライセンス利用状況を以下のコマンドで確認できます。

```
[login]$ lmutil lmstat -S SCHROD -c *****@kvm5.ini.t4.gsic.titech.ac.jp
```

Info

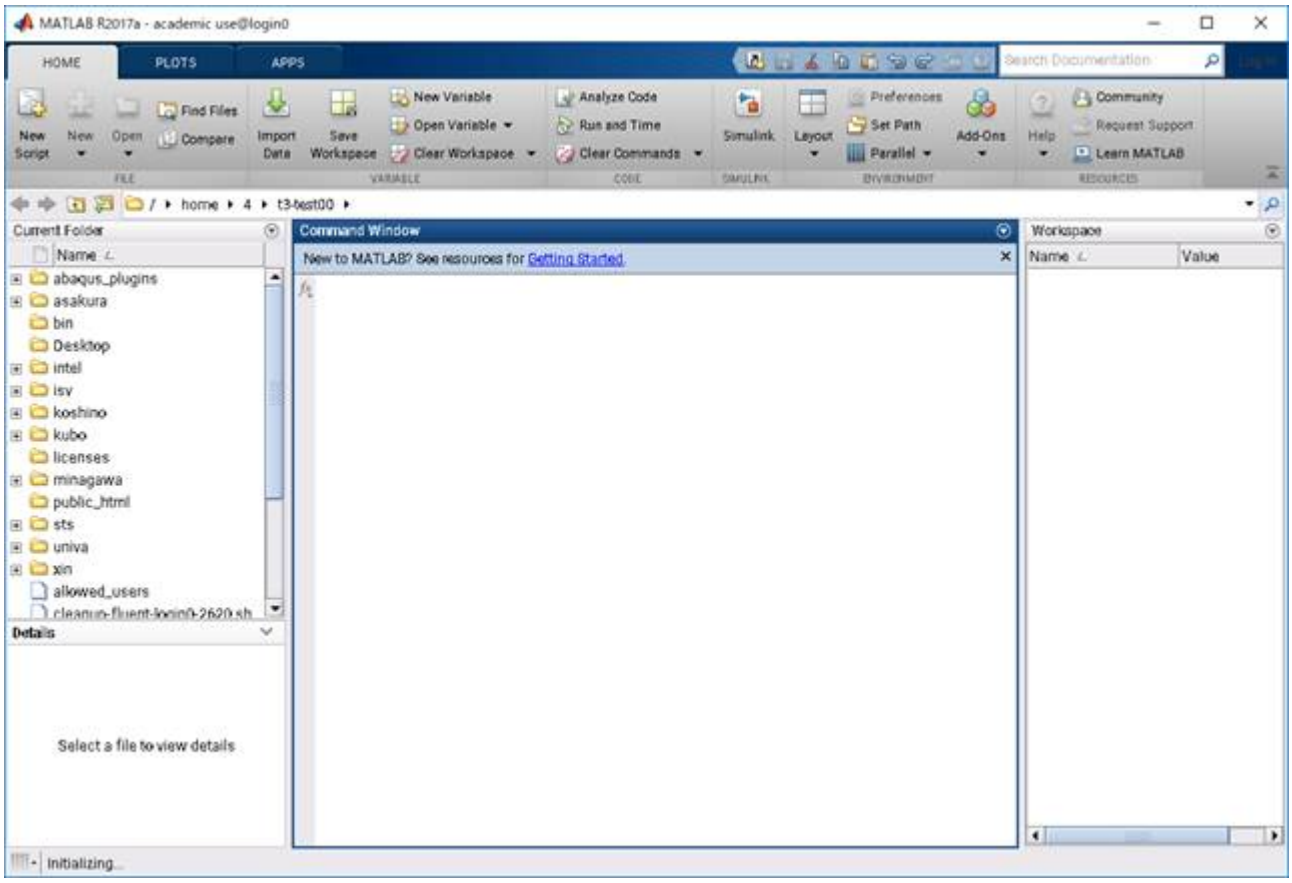
ポート番号はアプリケーション有効化(TSUBAME4)で購入後に参照可能です。
/apps/t4/rhel9/isv/schrodinger/t4-license
ポート番号は毎年4月中旬頃変更されます。

6.14. MATLAB

MATLABは行列計算などの数値計算やデータの可視化をすることのできるアプリケーションです。

MATLABの利用方法の例を以下に示します。

```
[rNnN]$ module load matlab
[rNnN]$ matlab
```



CLIでの使用手順について

```
[rNnN]$ module load matlab
[rNnN]$ matlab -nodisplay
```

終了するにはexitを入力します。

MATLABのライセンス利用状況を以下のコマンドで確認できます。

```
[login]$ lmutil lmstat -S MLM -c *****@kvm5.ini.t4.gsic.titech.ac.jp:****@kvm6.ini.t4.gsic.titech.ac.jp:*****@ldap2.ini.t4.gsic.titech.ac.jp
```

Info

ポート番号はアプリケーション有効化(TSUBAME4)で購入後に参照可能です。
/apps/t4/rhel9/isv/matlab/t4-license

6.15. VASP

Info

VASPの利用には別途利用者ごとにライセンス契約が必要です。

Info

VASPに関するお問い合わせについては、ライセンス条項の関係上TSUBAME4.0の相談窓口での対応は限られます。
TSUBAME4.0 FAQに挙がっていない問い合わせについては、VASP Forumの参照・投稿をご検討ください。

TSUBAME4.0でVASPを使用するには、2つの方法があります。

・コンパイル済みのバイナリを利用する
情報基盤センターでは、VASPライセンス保持者を対象にコンパイル済みのバイナリへのアクセスを提供しております。
VASP6の有効なライセンスをお持ちで、TSUBAME4.0においてコンパイル済バイナリを利用したい場合は、こちらの申請フォームへ登録をお願いします。
所定の手続きが完了次第、TSUBAME上のバイナリへのアクセス許可を設定させていただきます。
(ライセンス発行元への確認が必要となるため、時間がかかる場合があります。)
VASPのコンパイル済みバイナリ利用法を以下に示します。

```
[rNnN]$ module load VASP
[rNnN]$ vasp_std（もしくは vasp_ncl など）
```

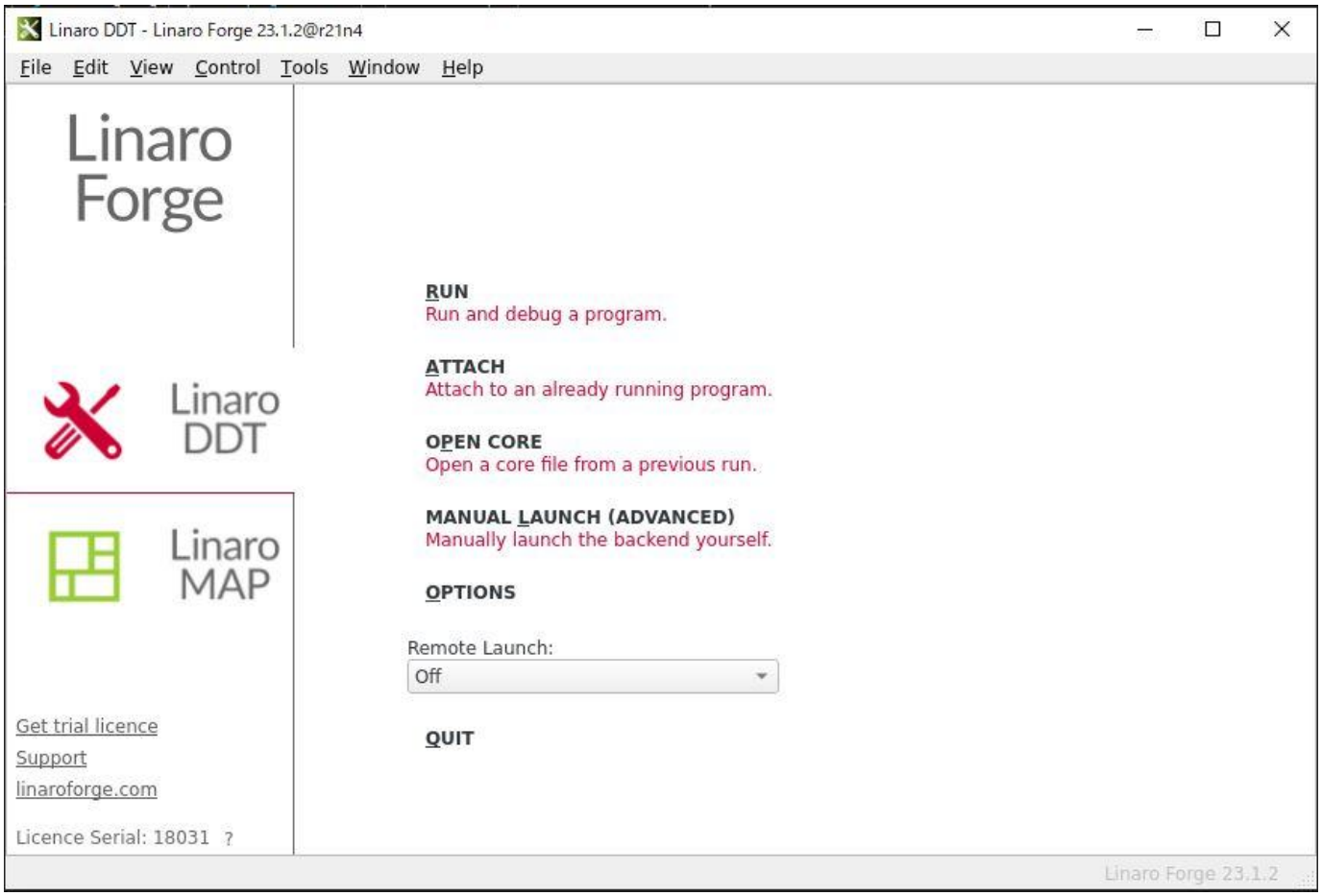
・ご自身でソースコードをビルドする
この場合、申請などは必要ありません。ご自身でソースコードを用意し、ビルドを実施してください。
また、moduleコマンドも不要です。ご自身の環境に合わせて環境変数などを指定してください。
参考情報として、TSUBAME4.0上でVASP6.4.2をビルドした際の手順を公開しています。

・TSUBAME4.0上でVASPをビルドした際の手順が知りたい。

6.16. Linaro forge(旧:Arm Forge)

Linaro forgeの利用方法を以下に示します。

```
[rNnN]$ module load forge
[rNnN]$ forge
```



メニューバーの File > Exit をクリックすると終了します。

7. フリーウェア

Info

本ページのコマンドライン例では、以下の表記を使用します。
[login]$: ログインノード
[rNnN]$: 計算ノード
[login/rNnN]$: ログインノードまたは計算ノード
[yourPC]$: ログインノードへの接続元環境

フリーウェアの一覧表を以下に示します。

ソフトウェア名	概要
GAMESS	ソルバ・シミュレータ
Tinker	ソルバ・シミュレータ
GROMACS	ソルバ・シミュレータ
LAMMPS	ソルバ・シミュレータ
NAMMD	ソルバ・シミュレータ
QUANTUM ESPRESSO	ソルバ・シミュレータ
CP2K	ソルバ・シミュレータ
OpenFOAM	ソルバ・シミュレータ、可視化
CUDA	GPUライブラリ
CuDNN	GPUライブラリ
NCCL	GPUライブラリ
TensorFlow	DeepLearningフレームワーク
DeePMD-kit	MD用DeepLearningフレームワーク
PyTorch	機械学習
R	インタプリタ(Rmpi,rpudに対応)
Hadoop	分散データ処理ツール
POV-Ray	可視化
ParaView	可視化
VisIt	可視化
vmd	可視化
VESTA	可視化
turbovnc	リモートGUI(X11) 表示
VirtualGL	リモートGUI
Open OnDemand	HPC向けWebポータル
gnuplot	データ可視化
GIMP	画像表示・編集
Tgif	画像表示・編集
ImageMagick	画像表示・編集
TeX Live	TeX ディストリビューション
OpenJDK	開発環境
python	開発環境
ruby	開発環境
perl	開発環境
PHP	開発環境
golang	開発環境

ソフトウェア名	概要
Emacs	エディタ
vim	エディタ
PETSc	リニアシステムソルバ、ライブラリ
FFTW	高速フーリエ変換ライブラリ
Apptainer	コンテナ管理
Spack	ソフトウェアパッケージ管理
miniconda	ソフトウェアパッケージ管理
PyPI	ソフトウェアパッケージ管理
Rbenv	ソフトウェアパッケージ管理
Alphafold	バイオ
tmux	端末多重接続
NetCDF	多次元データフォーマット
HDF5	階層データフォーマット
ffmpeg	動画・音声処理

Info

本ページで使用しているmoduleコマンドについては利用環境の切換え方法を参照してください。

7.1. 量子化学/MD関連ソフトウェア

7.1.1. GAMESS

GAMESSはオープンソースの第一原理分子量子化学計算アプリケーションです。

バッチキューシステムを利用したGAMESSの利用方法の例を以下に示します。

```
#!/bin/bash
#$ -cwd
#$ -l node_f=1
#$ -l h_rt=0:10:0
#$ -N gamess


module load gamess
$GAMESS_DIR/rungms exam01 00 2 2 1
```

詳細な説明は以下に記載されています。

http://www.msg.ameslab.gov/gamess/index.html

7.1.2. Tinker

Tinkerはバイオポリマーの為の特別な機能を備えた、分子動力学の為のモデリングソフトウェアです。

バッチキューシステムを利用したTinkerの利用方法の例を以下に示します。

```
#!/bin/bash
#$ -cwd
#$ -l node_f=1
#$ -l h_rt=0:10:0
#$ -N tinker
```

```
module load tinker
cp -rp $TINKER_DIR/example $T4TMPDIR
cd $T4TMPDIR/example
dynamic waterbox.xyz -k waterbox.key 100 1 1 2 300
cp -rp $T4TMPDIR/example $HOME
```

詳細な説明は以下に記載されています。

https://dasher.wustl.edu/tinker/

7.1.3. GROMACS

GROMACSは分子動力学シミュレーションとエネルギー最小化を行う為のエンジンです。

バッチキューシステムを利用したGROMACSの利用方法の例を以下に示します。

```
#!/bin/bash
#$ -cwd
#$ -l node_f=1
#$ -l h_rt=0:10:0
#$ -N gromacs


module load gromacs
cp -rp $GROMACS_DIR/examples/water_GMX50_bare.tar.gz $T4TMPDIR
cd $T4TMPDIR
tar xf water_GMX50_bare.tar.gz
cd water-cut1.0_GMX50_bare/3072
gmx_mpi grompp -f pme.mdp
OMP_NUM_THREADS=2 mpiexec -np 4 gmx_mpi mdrun
cp -rp $T4TMPDIR/water-cut1.0_GMX50_bare $HOME
```

詳細な説明は以下に記載されています。

http://www.gromacs.org/

Info

GROMACS 2023以降はCUDA Graphs機能に対応しています。 本機能を利用した場合、複数基のGPU使用時に実行性能の向上につながる可能性がありますが、以下の記述もあるためご自分のケースで高速化するかどうかは各自の責任でご判断ください。
「この機能はまだ実験的なものであり、テストも限られているため、結果が期待通りであることを確認するために注意が必要です」

CUDA Graphsについては以下のサイトをご参照ください。
https://developer.nvidia.com/ja-jp/blog/a-guide-to-cuda-graphs-in-gromacs-2023/

TSUBAME4.0 node_f=1 の場合の実行コマンドの一例
mpiexec -x OMP_NUM_THREADS=2 -x GMX_ENABLE_DIRECT_GPU_COMM=1 -np 4 gmx_mpi mdrun -nb gpu -bonded gpu -pme gpu -update gpu -npme 1

7.1.4. LAMMPS

LAMMPSは液状、固体状、気体状の粒子の集団をモデル化する古典分子動力学コードです。

バッチキューシステムを利用したLAMMPSの利用方法の例を以下に記します。

```
#!/bin/bash
#$ -cwd
#$ -l node_f=1
#$ -l h_rt=0:10:0
#$ -N lammps


module load lammps
cp -rp $LAMMPS_DIR/examples/VISCOSITY $T4TMPDIR
cd $T4TMPDIR/VISCOSITY
mpirun -x PATH -x LD_LIBRARY_PATH -np 4 lmp -sf gpu -in in.gk.2d
cp -rp $T4TMPDIR/VISCOSITY $HOME
```

詳細な説明は以下に記載されています。

http://lammps.sandia.gov/

7.1.5. NAMD

NAMDは、大規模な生体分子システムの高性能シミュレーション用にデザインされたオブジェクト指向の並列分子動力学コードです。

バッチキューシステムを利用したNAMDの利用方法の例を以下に記します。

```
#!/bin/bash
#$ -cwd
#$ -l node_f=1
#$ -l h_rt=0:10:0
#$ -N namd


module load namd
cp -rp $NAMD_DIR/examples/stmv.tar.gz $T4TMPDIR
cd $T4TMPDIR
tar xf stmv.tar.gz
cd stmv
namd3 +idlepoll +p4 +devices 0,1,2,3 stmv.namd
cp -rp $T4TMPDIR/stmv $HOME
```

詳細な説明は以下に記載されています。

https://www.ks.uiuc.edu/Research/namd/3.0/ug/

7.1.6. CP2K

CP2Kは固体、液体、分子、周期的、物質、結晶、生物系の原子シミュレーションを実行できる量子化学、固体物理ソフトウェアパッケージです。

バッチキューシステムを利用したCP2Kの利用方法の例を以下に記します。

```
#!/bin/bash
#$ -cwd
#$ -l node_f=1
#$ -l h_rt=0:10:0
#$ -N cp2k


module load cp2k
cp -rp $CP2K_DIR/benchmarks/QS $T4TMPDIR
cd $T4TMPDIR/QS
export OMP_NUM_THREADS=1
mpirun -x PATH -x LD_LIBRARY_PATH -np 4 cp2k.psmp -i H2O-32.inp -o H2O-32.out
cp -rp $T4TMPDIR/QS $HOME
```

詳細な説明は、以下に記載されています。

https://www.cp2k.org/

7.1.7. QUANTUM ESPRESSO

QUANTUM ESPRESSOは第一原理電子構造計算と材料モデリングのためのスイートです。
バッチキューシステムを利用したQUANTUM ESPRESSOの利用方法の例を以下に記します。

```
#!/bin/sh
#$ -cwd
#$ -l h_rt=00:10:00
#$ -l node_f=1
#$ -N q-e


module purge
module load quantumespresso

cp -p $QUANTUMESPRESSO_DIR/test-suite/pw_scf/scf.in .
cp -p $QUANTUMESPRESSO_DIR/example/Si.pz-vbc.UPF .

mpirun -x ESPRESSO_PSEUDO=$PWD -x PATH -x LD_LIBRARY_PATH -np 4 pw.x < scf.in
```

Info

TSUBAME4.0で導入しているQUANTUM ESPRESSOは、CPUに対応していません。必ずGPUを使用可能なリソースタイプを指定してください。
また、-npに指定する並列数は使用するリソースタイプのGPU数と一致させてください。(Round up)
例　node_f 使用時は -np 4 を指定。 gpu_h 使用時は -np 1 を指定。

また、"OpenMPI/Intel MPI実行時に、hcoll 関連のErrorやsegmentation faultが発生する。"対応を実施する場合、必ず1ノードで実行してください。
マルチノードで実行した場合、カーネルパニックが発生する可能性があります。

詳細な説明は、以下に記載されています。

https://www.quantum-espresso.org/

7.2. CFD関連ソフトウェア

7.2.1. OpenFOAM

OpenFOAMはオープンソースの流体/連続体シミュレーションコードです。
Foudation版(openfoam)とESI版(openfoam-esi)の2種類がインストールされています。
バッチキューシステムを利用したOpenFOAMの利用方法の例を以下に記します。

```
#!/bin/bash
#$ -cwd
#$ -l node_f=1
#$ -l h_rt=0:10:0
#$ -N openform


module load openfoam
mkdir -p $T4TMPDIR/$FOAM_RUN
cd $T4TMPDIR/$FOAM_RUN
cp -rp $FOAM_TUTORIALS .
cd tutorials/legacy/incompressible/icoFoam/cavity/cavity
blockMesh
icoFoam
paraFoam
```

ESI版OpenFOAMをご利用の場合は上記の `module load` の箇所を `module openfoam-esi` として下さい。

詳細な説明は以下に記載されています。

https://openfoam.org/resources/
http://www.openfoam.com/documentation/

7.3. GPU用数値計算ライブラリ

7.3.1. cuBLAS

cuBLASはGPUで動作するBLAS　Basic Linear Algebra Subprograms　ライブラリです。

利用方法

```
[rNnN]$ module load cuda
[rNnN]$ nvcc -gencode arch=compute_90,code=sm_90 -o sample sample.cu -lcublas
```

通常のC言語のプログラム中で、cuBLASを呼び出す場合、コンパイル時に-I、-L、-lで指定する必要があります。

```
[rNnN]$ module load cuda
[rNnN]$ gcc -o blas blas.c -I${CUDA_HOME}/include -L${CUDA_HOME}/lib64 -lcublas
```

7.3.2. cuSPARSE

cuSPARSEはNVIDIA GPU上で疎行列計算を行うためのライブラリです。

利用方法

```
[rNnN]$ module load cuda
[rNnN]$ nvcc -gencode arch=compute_90,code=sm_90 sample.cu -lcusparse -o sample
```

通常のC言語のプログラム中で、cuSPARSEを呼び出す場合、コンパイル時に-I、-L、-lで指定する必要があります。

```
[rNnN]$ module load cuda
[rNnN]$ g++ sample.c -lcusparse_static -I${CUDA_HOME}/include -L${CUDA_HOME}/lib64  -lculibos -lcudart_static -lpthread -ldl -o sample
```

7.3.3. cuFFT

cuFFTはNVIDIA GPU上で並列FFT(高速フーリエ変換)を行うためのライブラリです。

利用方法

```
[rNnN]$ module load cuda
[rNnN]$ nvcc -gencode arch=compute_90,code=sm_90 -o sample sample.cu -lcufft
```

通常のC言語のプログラム中で、cuFFTを呼び出す場合、コンパイル時に-I、-L、-lで指定する必要があります。

```
[rNnN]$ module load cuda
[rNnN]$ gcc -o blas blas.c -I${CUDA_HOME}/include -L${CUDA_HOME}/lib64 -lcufft
```

7.4. 機械学習、ビックデータ解析関連ソフトウェア

7.4.1. CuDNN

CuDNNはGPUを用いたDeep Neural Networkの為のライブラリです。

CuDNNの利用方法を以下に記します。

```
[rNnN]$ module load cuda cudnn
```

7.4.2. NCCL

NCCLは複数GPUの為の集団通信ライブラリです。

NCCLの利用方法の例を以下に記します。

```
[rNnN]$ module load cuda nccl
```

7.4.3. PyTorch

PyTorch は Python で機械学習を行う場合に利用できるオープンソースの機械学習ライブラリです。

PyTorchのインストール方法を以下に記します。

```
[rNnN]$ python3 -m pip install --user torch
```

Info

PyTorch はユーザ環境にインストールします。

7.4.4. TensorFlow

TensorFlowはデータフローグラフを用いた機械学習・AIのオープンソースのライブラリです。

TensorFlowのインストール方法を以下に記します。

```
[rNnN]$ python3 -m pip install --user tensorflow
```

Info

TensorFlow はユーザ環境にインストールします。

詳細な説明は以下に記載されています。

https://www.tensorflow.org/

Info

TensorFlowをGPU上で動作させる場合、python,cudnn,cudaのバージョンを下記リンク先の表に合わせる必要があります。

https://www.tensorflow.org/install/source#gpu

例
TensorFlow 2.17.0の場合
Python 3.9　3.12
CuDNN 8.9
CUDA 12.3

なお、TSUBAME4.0ではCUDA12に対応したCuDNN 8.9は導入しておりません。
また、原則として現在導入されているバージョンよりも古いバージョンの導入は行いません。
TensorFlow 2.17.0をGPU上で動作させたい場合、pipなどを使用してご自身の環境にCuDNN 8.9を導入してください。
(CuDNNを使用しない、またはCuDNN9.0.0を使用した場合について、いずれもGPU上で動作しないことを確認しました。)

また、構築環境や組み合わせによっては、条件を満たしていても正常に導入が出来ないケースがあるようです。
こちらで検証した手順について、参考情報として以下に記載します。
それ以外のバージョン・手順・環境で構築した場合のトラブルについては、ご自身で各アプリのリリースノートなどを確認し解決いただく必要があります。

[参考情報]
python仮想環境にてTensorFlow 2.17.0とCUDA12に対応したCuDNN 8.9を導入してGPUの利用を確認するまでのログを掲載します。

```
# pythonバージョンの確認
[rNnN]$ python -V
Python 3.9.18

# cuda12.3.2の読込
[rNnN]$ module load cuda/12.3.2

# python仮想環境の作成
[rNnN]$ python -m venv venv

# python仮想環境の呼出
[rNnN]$ source venv/bin/activate

# pipのバージョンアップ
[rNnN]$ pip install --upgrade pip

# cuda12対応のcudnn8.9とtenserflow2.17.0のインストール
[rNnN]$ pip install nvidia-cudnn-cu12==8.9.7.29 tensorflow==2.17.0

# GPUが見えているかの確認
[rNnN]$ python3 -c "import tensorflow as tf; print(tf.config.list_physical_devices('GPU'))"
<< ログ省略 >>
[PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU')]
```

7.4.5. DeePMD-kit

DeePMD-kitはMD用機械学習フレームワークです。
DeePMD-kitのジョブスクリプトの例を以下に記します。

7.4.5.1. 1 DeePMD-kit + LAMMPS

7.4.5.1.1. DEEPMD-KIT LAMMPS 1ノード

DeePMD-kit + LAMMPSのジョブスクリプト例(1ノード、4GPU)を以下に示します。

```
#!/bin/sh
#$ -l h_rt=6:00:00
#$ -l node_f=1
#$ -cwd


module purge
module load deepmd-kit lammps
module li 2>&1

# enable DeePMD-kit for lammps/2aug2023_u3
export LAMMPS_PLUGIN_PATH=$DEEPMD_KIT_DIR/lib/deepmd_lmp

# https://tutorials.deepmodeling.com/en/latest/Tutorials/DeePMD-kit/learnDoc/Handson-Tutorial%28v2.0.3%29.html

wget https://dp-public.oss-cn-beijing.aliyuncs.com/community/CH4.tar
tar xf CH4.tar

cd CH4/00.data
python3 <<EOF
import dpdata
import numpy as np
data = dpdata.LabeledSystem('OUTCAR', fmt = 'vasp/outcar')
print('# the data contains %d frames' % len(data))
# random choose 40 index for validation_data
index_validation = np.random.choice(200,size=40,replace=False)
# other indexes are training_data
index_training = list(set(range(200))-set(index_validation))
data_training = data.sub_system(index_training)
data_validation = data.sub_system(index_validation)
# all training data put into directory:"training_data"
data_training.to_deepmd_npy('training_data')
# all validation data put into directory:"validation_data"
data_validation.to_deepmd_npy('validation_data')
print('# the training data contains %d frames' % len(data_training))
print('# the validation data contains %d frames' % len(data_validation))
EOF


cd ../01.train
dp train input.json
dp freeze -o graph.pb
dp compress -i graph.pb -o graph-compress.pb
dp test -m graph-compress.pb -s ../00.data/validation_data -n 40 -d results

cd ../02.lmp
ln -s ../01.train/graph-compress.pb
mpirun -x PATH -x LD_LIBRARY_PATH -np 4 lmp -sf gpu -in in.lammps
```

7.4.5.1.2. DEEPMD-KIT LAMMPS 2ノード

DeePMD-kit + LAMMPSのジョブスクリプト例(2ノード、8GPU)を以下に示します。

```
#!/bin/sh
#$ -l h_rt=12:00:00
#$ -l node_f=2
#$ -cwd


module purge
module load deepmd-kit lammps
module li 2>&1

# enable DeePMD-kit
export LAMMPS_PLUGIN_PATH=$DEEPMD_KIT_DIR/lib/deepmd_lmp

# https://tutorials.deepmodeling.com/en/latest/Tutorials/DeePMD-kit/learnDoc/Handson-Tutorial%28v2.0.3%29.html

wget https://dp-public.oss-cn-beijing.aliyuncs.com/community/CH4.tar
tar xf CH4.tar

cd CH4/00.data
python3 <<EOF
import dpdata
import numpy as np
data = dpdata.LabeledSystem('OUTCAR', fmt = 'vasp/outcar')
print('# the data contains %d frames' % len(data))
# random choose 40 index for validation_data
index_validation = np.random.choice(200,size=40,replace=False)
# other indexes are training_data
index_training = list(set(range(200))-set(index_validation))
data_training = data.sub_system(index_training)
data_validation = data.sub_system(index_validation)
```

```
# all training data put into directory:"training_data"
data_training.to_deepmd_npy('training_data')
# all validation data put into directory:"validation_data"
data_validation.to_deepmd_npy('validation_data')
print('# the training data contains %d frames' % len(data_training))
print('# the validation data contains %d frames' % len(data_validation))
EOF

cd ../01.train
mpirun -x PATH -x LD_LIBRARY_PATH -x PYTHONPATH -x NCCL_BUFFSIZE=1048576 -npernode 4 -np 8 dp train input.json
dp freeze -o graph.pb
dp compress -i graph.pb -o graph-compress.pb
dp test -m graph-compress.pb -s ../00.data/validation_data -n 40 -d results

cd ../02.lmp
ln -s ../01.train/graph-compress.pb
mpirun -x PATH -x LD_LIBRARY_PATH  -npernode 4 -np 8 lmp -sf gpu -in in.lammps
```

詳細な説明は以下に記載されています。
https://docs.deepmodeling.com/projects/deepmd/en/master/index.html

7.4.6. R

Rはデータ解析とグラフィックスの為のインタプリタ型プログラミング言語です。

並列処理用にRmpi、GPU用にrpudがインストールされています。

Rの利用方法の例を以下に記します。

```
[rNnN]$ module load R
[rNnN]$ mpirun -np 2 Rscript test.R
```

7.4.7. Apache Hadoop

Apache Hadoopソフトウェアライブラリは単純なプログラミングモデルを用いて大きなデータセットを分散処理する為のフレームワークです。

Apache Hadoopの利用方法の例を以下に記します。

```
[rNnN]$ module load hadoop
[rNnN]$ mkdir input
[rNnN]$ cp -p $HADOOP_HOME/etc/hadoop/*.xml input
[rNnN]$ hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.6.jar grep input output 'dfs[a-z.]+'
[rNnN]$ cat output/part-r-00000
1       dfsadmin
```

バッチキューシステムの場合の利用手順を以下に示します。

```
#!/bin/bash
#$ -cwd
#$ -l node_f=1
#$ -l h_rt=0:10:0
#$ -N hadoop


module load hadoop
cd $T4TMPDIR
mkdir input
cp -p $HADOOP_HOME/etc/hadoop/*.xml input
hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.6.jar grep input output 'dfs[a-z.]+'
cp -rp output $HOME
```

7.5. 可視化関連ソフトウェア

7.5.1. POV-Ray

POV-Rayはフリーの光線追跡ソフトです。

POV-Rayの利用方法の例を以下に記します。

```
[rNnN]$ module load pov-ray
[rNnN]$ povray -benchmark
```

詳細な説明は以下に記載されています。

http://www.povray.org/

7.5.2. ParaView

ParaViewはオープンソース、マルチプラットフォームのデータ解析と可視化アプリケーションです。

ParaViewの利用方法の例をを以下に記します。

```
[rNnN]$ module load paraview
[rNnN]$ paraview
```

7.5.2.1. 複数GPUを用いて可視化する場合

`paraview/5.12.0`、 `paraview/5.12.0-egl`、を用いて複数ノードで複数GPUを用いて可視化することができます。
`paraview/5.12.0-egl`には`paraview`コマンドが含まれていないことにご注意ください。
以下は`node_f=2`で8GPUを使う例です。

・wrap.sh

```
#!/bin/sh

num_gpus_per_node=4
mod=$((OMPI_COMM_WORLD_RANK%num_gpus_per_node))

if [ $mod -eq 0 ];then
    export VTK_DEFAULT_EGL_DEVICE_INDEX=0
elif [ $mod -eq 1 ];then
    export VTK_DEFAULT_EGL_DEVICE_INDEX=1
elif [ $mod -eq 2 ];then
    export VTK_DEFAULT_EGL_DEVICE_INDEX=2
elif [ $mod -eq 3 ];then
    export VTK_DEFAULT_EGL_DEVICE_INDEX=3
fi

$*
```

・job.sh

```
#!/bin/sh
#$ -cwd
#$ -V
#$ -l h_rt=8:0:0
#$ -l node_f=2

module purge
module load paraview

mpirun -x PATH -x LD_LIBRARY_PATH -npernode 4 -np 8 ./wrap.sh pvserver
```

`wrap.sh`に実行権限を忘れずに付与して下さい。(`chmod 755 wrap.sh`)
上記のジョブスクリプトで

```
[login]$ qsub -g <グループ名> job.sh
```

を行い、ジョブを投入します。
`qstat`でジョブが流れているのを確認します。

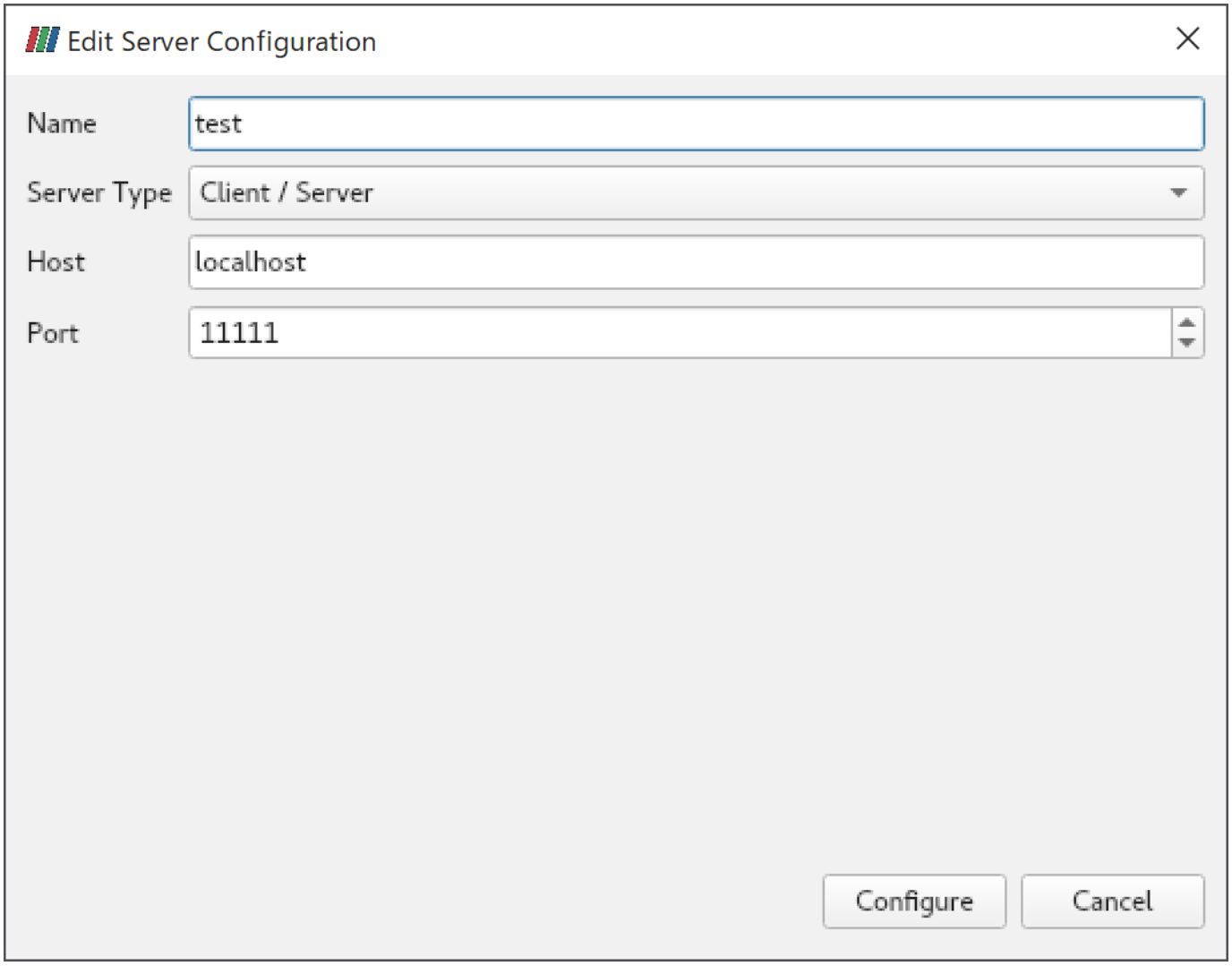
```
[login]$ qstat
job-ID     prior   name       user         state submit/start at     queue                          jclass                         slots ja-task-ID
-----------------------------------------------------------------------------------------------------------------------------------------------
   xxxxxxx 0.55354 job.sh     yyyyyyyy     r     05/31/2024 09:24:19 all.q@rXnY                                                 56
```

ジョブが流れているノードにX転送でsshし、paraviewを起動します。

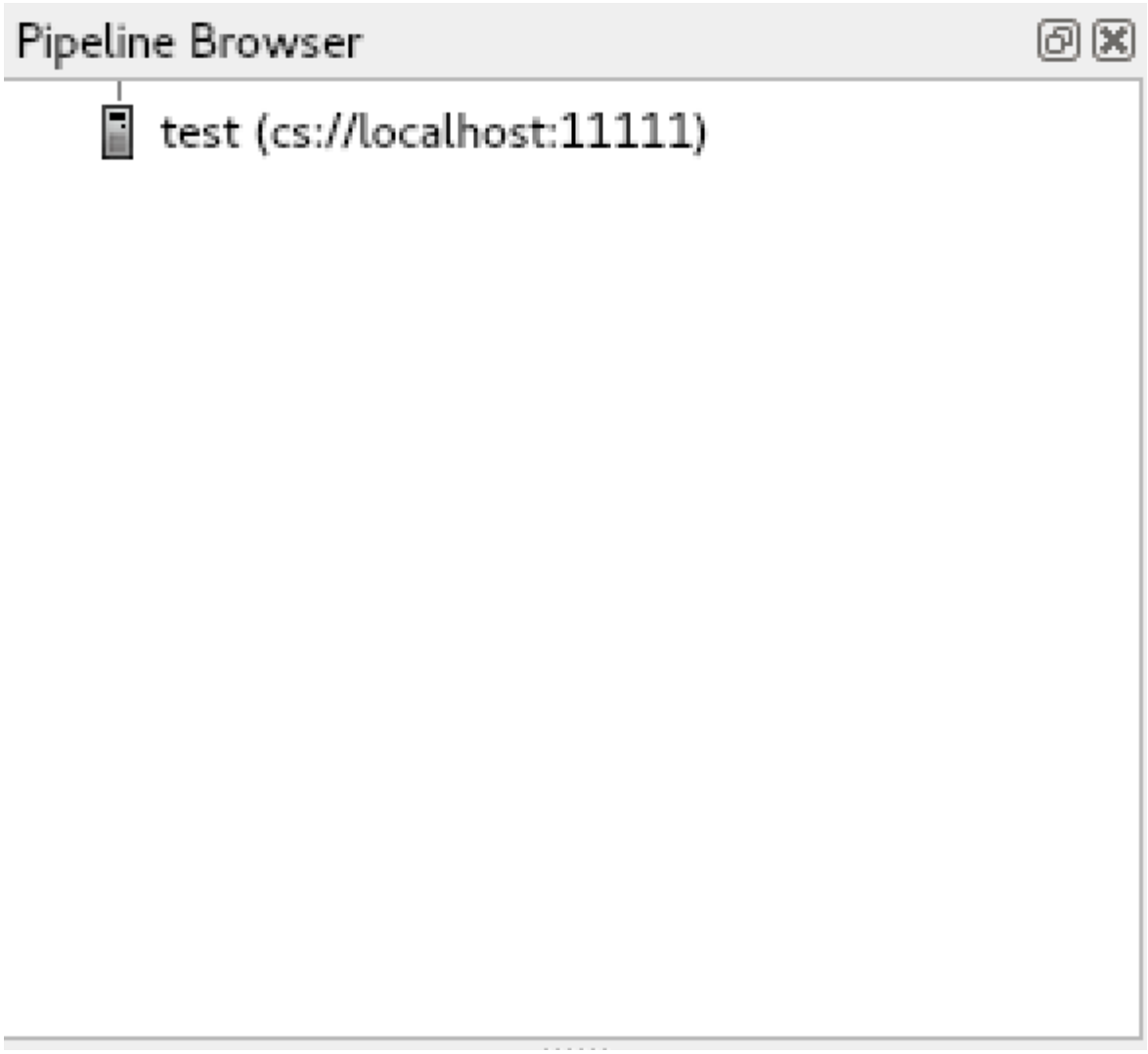
```
[login]$ ssh -CY <ジョブが流れている計算ノード>
```

```
[rNnN]$ module load paraview
[rNnN]$ paraview
```

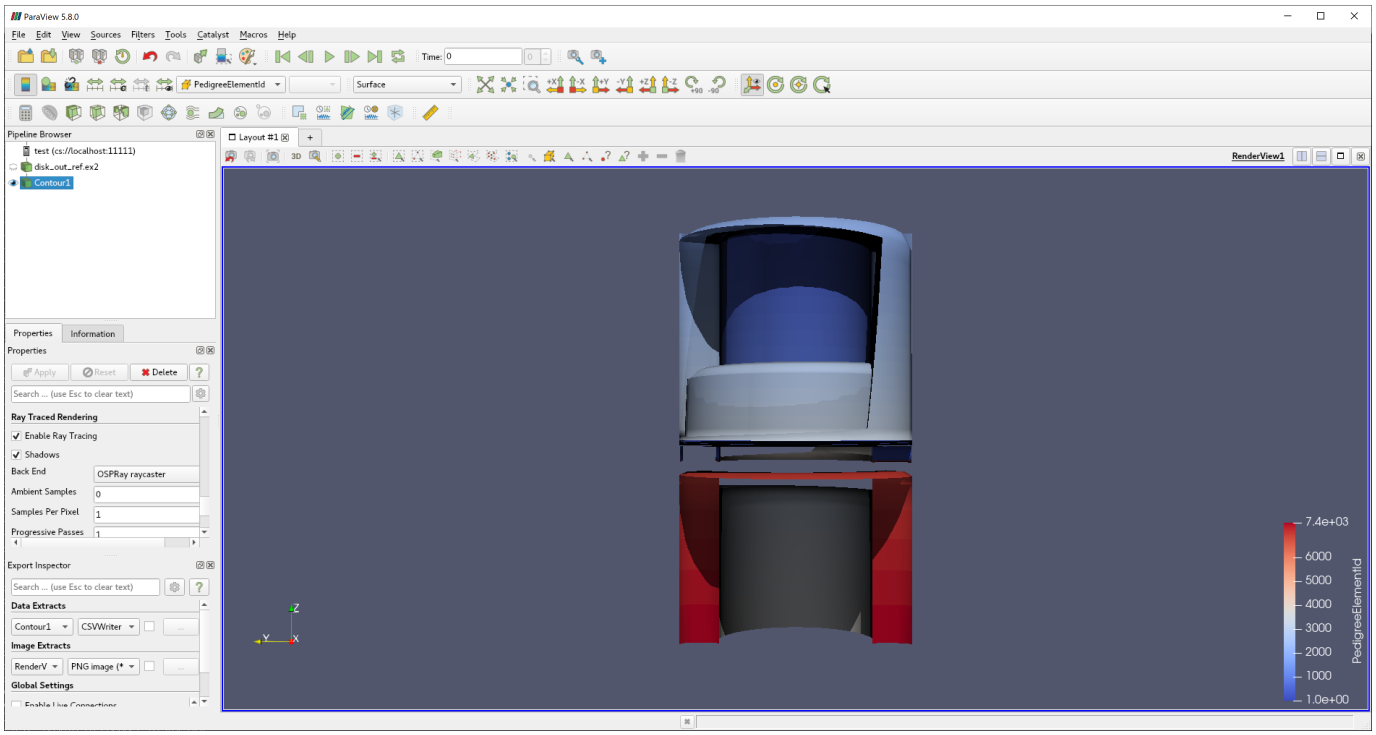
※turbovncを用いても可能です。
起動後、「File」->「Connect」をクリックし、「Add Server」をクリックします。
「Name」を適当に入力し(ここでは"test"とします)、「Configure」をクリックします。



その後、「Connect」をクリックします。
接続されると、「Pipeline Browser」の項目に`test(cs://localhost:11111)`が表示されます。



paraviewのサンプルデータはここからダウンロードすることができます。



詳細な説明は以下に記載されています。

https://www.paraview.org/

7.5.3. VisIt

VisItはオープンソースの可視化アプリケーションです。

VisItの利用方法の例を以下に記します。

```
[rNnN]$ module visit
[rNnN]$ visit
```

詳細な説明は以下に記載されています。

https://wci.llnl.gov/simulation/computer-codes/visit/

7.6. その他フリーウェア

7.6.1. turbovnc

turbovncはオープンソースのVNCソフトウェアです。
turbovncの使用方法の例を以下に記します。 ※qrshで計算ノードを確保し計算ノード上で実行して下さい。

・計算ノードを確保する

```
[login]$ qrsh -g <グループ名> -l <資源タイプ>=<個数> -l h_rt=<時間>
```

・確保した計算ノード上で以下を実行し、vncserverを起動する

```
[rNnN]$ module load turbovnc
[rNnN]$ vncserver -xstartup xfce.sh

You will require a password to access your desktops.

Password:  # <-- パスワードを聞かれるので設定する
Verify:
Would you like to enter a view-only password (y/n)? n
```

```
Desktop 'TurboVNC: rXnY:1 ()' started on display rXnY:1 # <-- ここのVNCのディスプレイ番号:1を覚えておく

Creating default startup script /home/n/xxxx/.vnc/xstartup.turbovnc
Starting applications specified in /home/n/xxxx/.vnc/xstartup.turbovnc
Log file is /home/n/xxxx/.vnc/rXiYnZ:1.log
```

画面サイズを大きくしたい場合は`vncserver -geometry <WIDTH>x<HEIGHT>`としてサイズを指定します。

- その後https://sourceforge.net/projects/turbovnc/files/から自分のPC用のインストーラをダウンロードし、turbovnc viewerをインストールする
- 計算ノードに接続したターミナルソフトからSSHポート転送の設定でローカルのポート5901を計算ノードのポート5901にポート転送するように設定する(もしディスプレイ番号がrXiYnZ:nだった場合、ポート転送のポート番号は5900+nに設定する)
- 自分のPCからturbovnc viewerを起動し、localhost:5901に接続して設定したパスワードを入力する

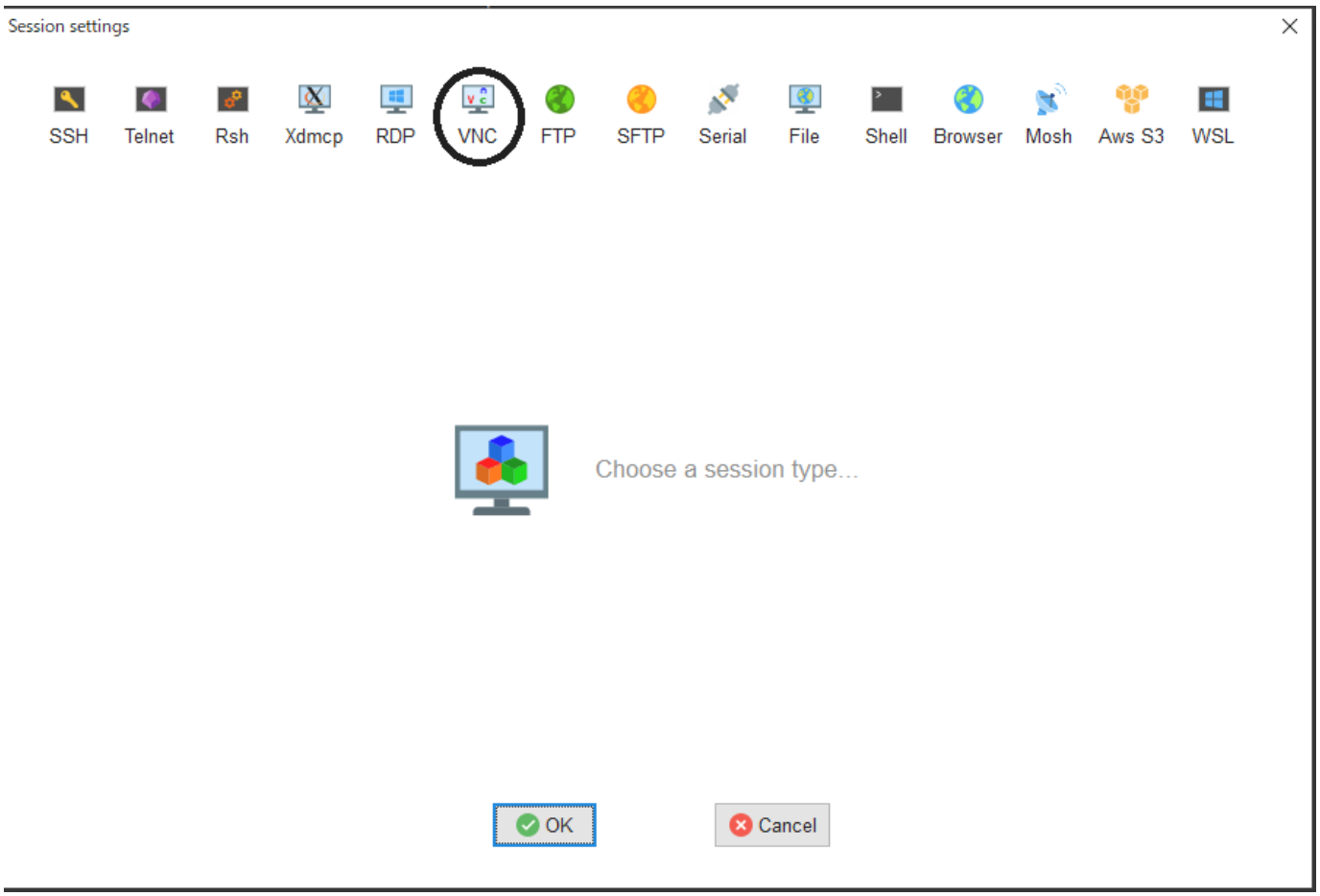
Tips

VNCのディスプレイ番号は vncserver が起動されるごとにカウントアップしていきます。SSHポート転送の設定のポート番号は毎回確認してください。

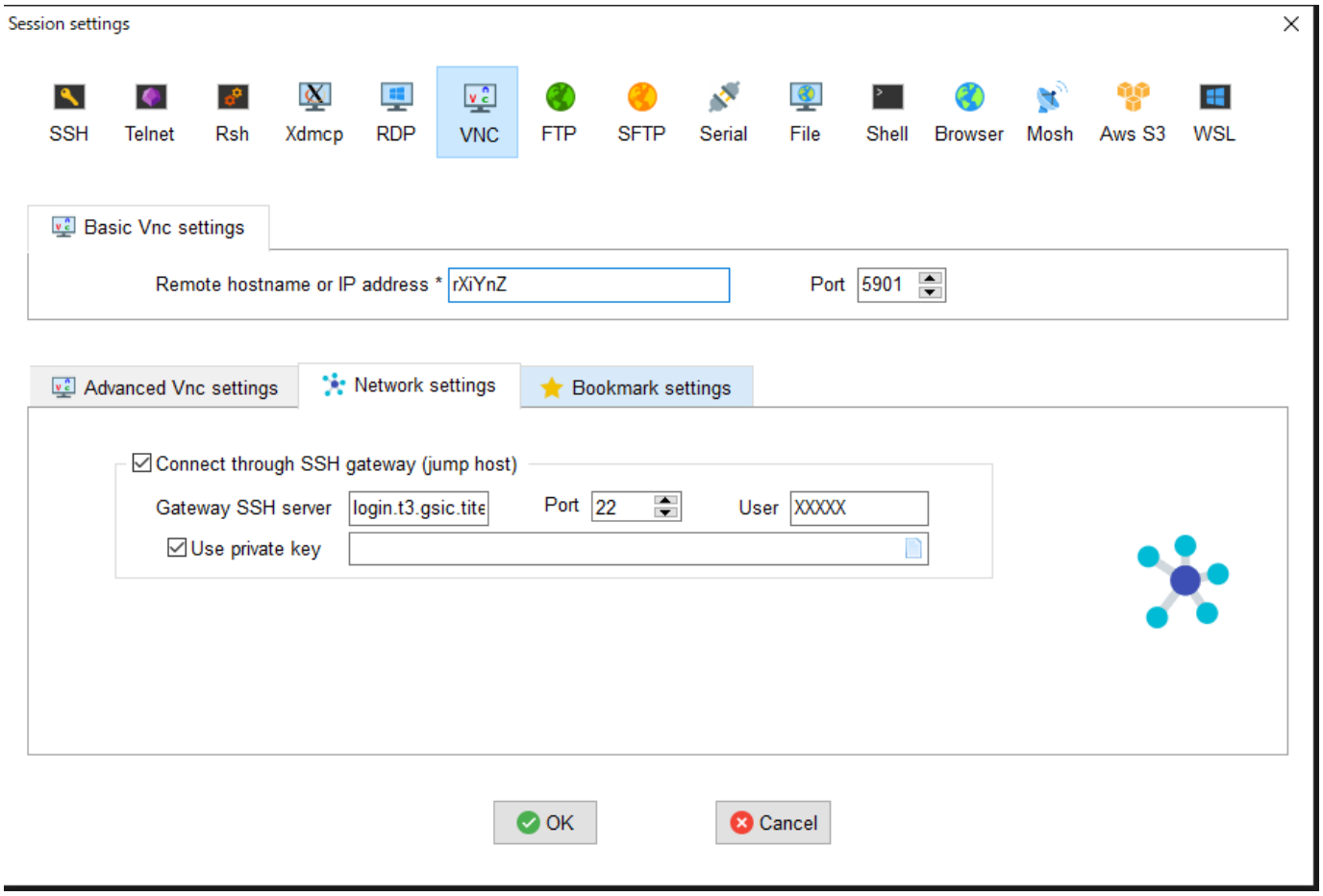
7.6.1.1. MobaXtermからVNCクライアントを利用する方法

MobaXtermにはVNCクライアントが内蔵されておりますので、VNCクライアントをインストールしなくてもVNC接続がご利用できます。

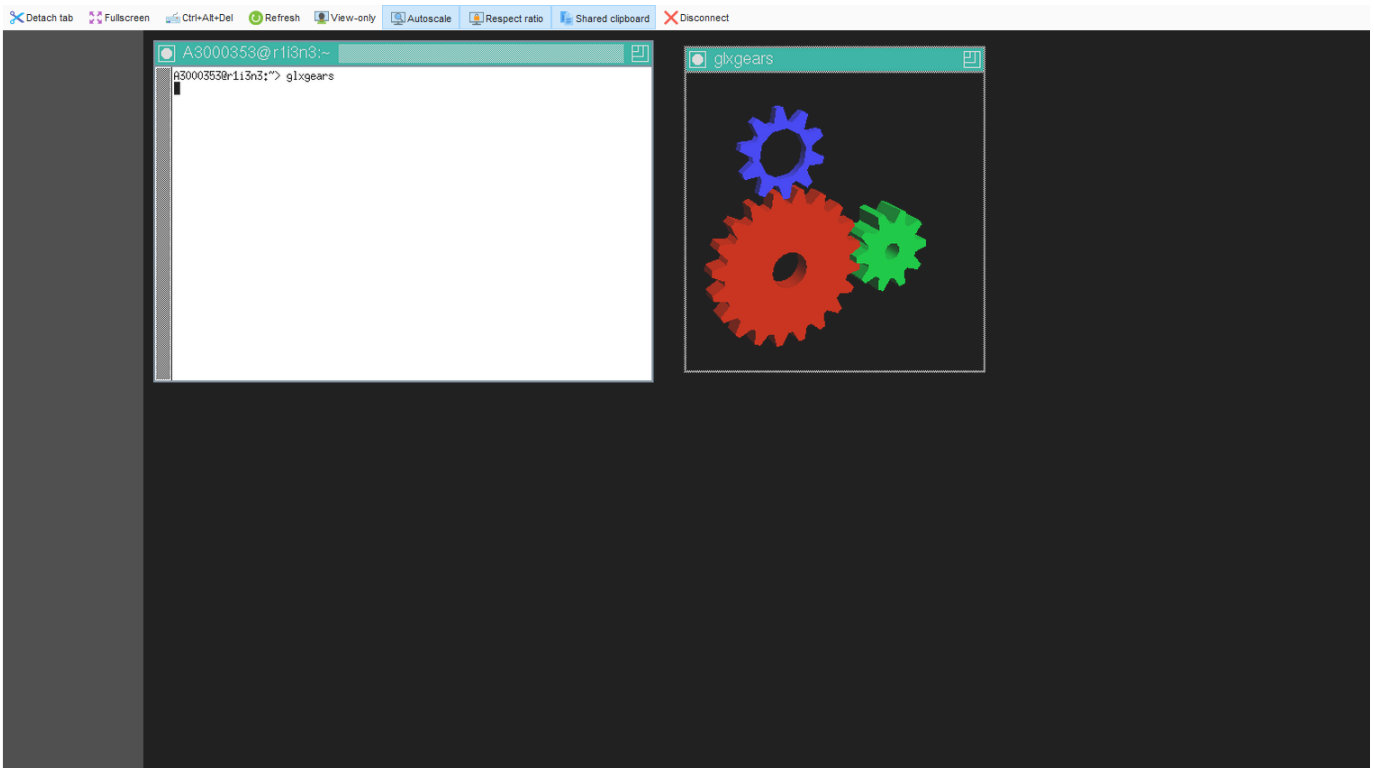
- qrshでノードを確保後、MobaXtermから「Sessions」->「New session」->「VNC」を選択する。



- その後、「Basic Vnc settings」の「Remote hostname or IP address」に確保した計算ノードのホスト名、「Port」を5900+nを入力し、「Network settings」の「Connect through SSH gateway(jump host)」をクリックし「Gateway SSH server」にlogin.t4.gsic.titech.ac.jpを入力、「Port」は22のまま、「User」に自分のTSUBAMEのログイン名を入力、「Use private key」にチェックを入れ自分の秘密鍵を入力する。



OKをクリックするとVNCクライアントが起動します。



7.6.1.2. turbovnc + VirtualGL

turbovnc用時に、GPUを1つ以上確保する資源タイプ(node_f, node_h, node_q, gpu_1)を用いている場合、VirtualGLを使用してGPUを用いて可視化することができます。
例として、gpu_1の場合のVirtualGLの使用例を以下に示します。

```
[login]$ qrsh <省略> -l gpu_1=1
[rNnN]$ module load turbovnc
[rNnN]$ vncserver -xstartup xfce.sh
```

・VNCクライアントで接続し、以下を実行

```
[VNCクライアント]$ vglrun -d /dev/dri/card<N> <OpenGLアプリケーション>
```

`<N>`は1　4の番号で、GPU0　3に対応します。
GPU番号と`card<N>`の対応は以下です。

デバイス名	**GPU番号**
/dev/dri/card1	GPU1(64:00)
/dev/dri/card2	GPU0(04:00)
/dev/dri/card3	GPU3(E4:00)
/dev/dri/card4	GPU2(84:00)

7.6.2. gnuplot

gnuplotはコマンドラインのインタラクティブなグラフ描画プログラムです。

標準機能に加え、X11、latex、PDFlib-lite、Qt4に対応するようにビルドされています。

gnuplotの利用方法の例を以下に記します。

```
[rNnN]$ gnuplot
```

7.6.3. Tgif

tgifはオープンソースの描画ツールです。

tgifの利用方法を以下に記します。

```
[rNnN]$ module load tgif
[rNnN]$ tgif
```

※Cannot open the Default(Msg) Font '*--courier-medium-r-normal*--14-----*-iso8859-1'.というエラーが出て起動しない場合は、 ~/.Xdefaultsに以下の行を追加して下さい。

```
Tgif.DefFixedWidthFont:          -*-fixed-medium-r-semicondensed--13-*-*-*-*-*-*-*
Tgif.DefFixedWidthRulerFont:     -*-fixed-medium-r-semicondensed--13-*-*-*-*-*-*-*
Tgif.MenuFont:                   -*-fixed-medium-r-semicondensed--13-*-*-*-*-*-*-*
Tgif.BoldMsgFont:                -*-fixed-medium-r-semicondensed--13-*-*-*-*-*-*-*
Tgif.MsgFont:                    -*-fixed-medium-r-semicondensed--13-*-*-*-*-*-*-*
```

7.6.4. GIMP

GIMPはオープンソースの画像操作プログラムです。

GIMPの利用方法の例を以下に記します。

```
[rNnN]$ gimp
```

7.6.5. ImageMagick

ImageMagickは画像処理ツールです。

標準機能に加え、X11、HDRI、libwmf、jpegに対応するようにビルドされています。

ImageMagickの利用方法の例を以下に記します。

```
[rNnN]$ module load imagemagick
[rNnN]$ convert -size 48x1024 -colorspace RGB 'gradient:#000000-#ffffff' -rotate 90 -gamma 0.5 -gamma 2.0 result.jpg
```

7.6.6. Tex Live

TeX Live は TeX の統合パッケージです。

TeX Live の利用方法の例を以下に記します。

```
[rNnN]$ lualatex test.tex
```

Info

PDFファイルが作成されます。

7.6.7. Java SDK

Java SDKとして、以下のバージョンのOpenJDKがインストールされています。
- openjdk version "1.8.0_402"
- openjdk version "11.0.22" (デフォルト)
- openjdk version "21.0.2"

module コマンドを利用して利用するバージョンを切り替えることが可能です。
切り替え方法は以下の通りです。

```
[login/rNnN]$ module unload openjdk
[login/rNnN]$ module load openjdk/バージョン指定
```

OpenJDKのバージョン	バージョン指定
1.8.0_402	1.8.0
11.0.22	11.0.22
21.0.2	21.0.2

現在のバージョンは、以下の手順で確認できます。

```
[login/rNnN]$ java -version
```

Java SDKの利用方法の例を以下に記します。

```
[login/rNnN]$ javac Test.java
[login/rNnN]$ java Test
```

7.6.8. PETSc

PETScはオープンソースの並列数値計算ライブラリです。線型方程式の求解等を行うことができます。

実数用、複素数用の2種類がインストールされています。

PETScの利用方法の例を以下に記します。

実数用

```
[rNnN]$ module load petsc/3.20.4-real
[rNnN]$ mpiifort test.F -lpetsc
```

複素数用

```
[rNnN]$ module load petsc/3.20.4-complex
[rNnN]$ mpiifort test.F -lpetsc
```

7.6.9. FFTW

FFTWはオープンソースの高速フーリエ変換用ライブラリです。

FFTWの利用方法の例を以下に記します。

Intel MPIの場合

```
[rNnN]$ module load fftw/3.3.10-intel intel-mpi/2021.11
[rNnN]$ gfortran test.f90 -lfftw3
```

Open MPIの場合

```
[rNnN]$ module load fftw/3.3.10-gcc
[rNnN]$ gfortran test.f90 -lfftw3
```

7.6.10. Apptainer (旧:Singularity)

Apptainer (旧:Singularity) は HPC 向け Linux コンテナです。
Apptainer の使い方については、コンテナの利用をご参照ください。

7.6.11. Alphafold2

Alphafold2は機械学習を用いたタンパク質構造予測プログラムです。
Alphafold2を利用する例を以下に示します。

・初期設定 (ログインノードもしくは計算ノード)

```
[login/rNnN]$ module purge
[login/rNnN]$ module load alphafold

[login/rNnN]$ cp -pr $ALPHAFOLD_DIR .
[login/rNnN]$ cd alphafold
```

・実行時 (alphafold/2.3.2のジョブスクリプトの例)

```
#!/bin/sh
#$ -l h_rt=24:00:00
#$ -l node_f=1
#$ -cwd


module purge
module load alphafold
module li

cd alphafold
./run_alphafold.sh -a 0,1,2,3 -d $ALPHAFOLD_DATA_DIR -o dummy_test/ -m model_1 -f ./example/query.fasta -t 2020-05-14
```

データベースファイルの容量が大きいため、可能な限り個別にダウンロードすることはお避け下さい。

Info

データベースファイルのみ利用したい場合、以下をご参照ください。
Alphafold2用データベース
module load alphafold を指定した場合、自動的にTSUBAME4.0内の最新データベースも指定されます。

Alphafoldの詳細な説明は以下をご参照下さい。
https://github.com/deepmind/alphafold

7.6.12. miniconda

minicondaはpythonの仮想環境作成ソフトウェアです。
minicondaを使用する例を以下に示します。

```
module load miniconda
eval "$(/apps/t4/rhel9/free/miniconda/24.1.2/bin/conda shell.bash hook)"
conda create -n test
conda acivate test
```

minicondaの詳細な説明は以下をご参照下さい。
https://docs.anaconda.com/free/miniconda/index.html

7.6.13. spack

spackはHPC向けのパッケージマネージャです。
spackを使用する例を以下に示します。

```
[rNnN]$ module load spack
[rNnN]$ spack install tree
```

spackの詳細は以下をご参照下さい。
https://spack.io/

7.6.14. Automake

Automakeは、コンパイルプロセスの一部を自動化するためのプログラミングツールです。
moduleコマンドで利用することで、別のバージョンのAutomakeを利用できます。

```
[login/rNnN]$ automake --version
automake (GNU automake) 1.16.2
```

```
[login/rNnN]$ module load automake
[login/rNnN]$ automake --version
automake (GNU automake) 1.17
```

Automakeの詳細は以下をご参照下さい。
https://www.gnu.org/software/automake/

7.6.15. Autoconf

Autoconfは、Bourneシェルが利用可能なコンピューターシステムでソフトウェアを構築、インストール、およびパッケージ化するための構成スクリプトを作成するためのツールです。
moduleコマンドで利用することで、別のバージョンのAutoconfを利用できます。

```
[login/rNnN]$ autoconf --version
autoconf (GNU Autoconf) 2.69
```

```
[login/rNnN]$ module load autoconf
[login/rNnN]$ autoconf --version
autoconf (GNU Autoconf) 2.72
```

Autoconfの詳細は以下をご参照下さい。
https://www.gnu.org/software/autoconf/

7.7. ソフトウェア用データベース

一部のソフトウェアで利用するデータベースをTSUBAME上に用意しました。これらのデータベースファイルは容量が大きいため、可能な限り個別にダウンロードすることはお避け下さい。

7.7.1. Alphafold2用データベース

Alphafold2で利用可能なデータベースです。

```
[rNnN]$ module load alphafold2_database
```

を実行することにより、環境変数**ALPHAFOLD_DATA_DIR**にデータベースへのパスが格納されます。
TSUBAMEで用意したAlphafold2を利用する場合はデータベースも自動で設定されますので、alphafold2_databaseを指定する必要はありません。
データベースは月1回程度の頻度で更新を予定しています。上記の指定を行うことで、常にTSUBAME4.0内の最新データベースが指定されます。
特定のタイミングのデータベースで固定したい場合、module load 時にバージョン指定をしてください。

[バージョン確認手順]

```
[login/rNnN]$ module load alphafold2_database
[login/rNnN]$ module list
Currently Loaded Modulefiles:
 1) alphafold2_database/202411
```

7.7.2. Alphafold3用データベース

Alphafold3で利用可能なデータベースです。

```
[rNnN]$ module load alphafold3_database
```

を実行することにより、環境変数**ALPHAFOLD_DATA_DIR**にデータベースへのパスが格納されます。
Alphafold3の実行環境はご自身で用意いただく必要があります。
東京科学大学 総合研究院 難治疾患研究所 森脇由隆先生が執筆されたQiitaにてTSUBAME4.0上での利用方法などが紹介されているため、こちらを参考にしてください。

Info

AlphaFold3のソースコードおよびモデルパラメータには、それぞれ異なるライセンスが適用されています。
商用利用の禁止など制限事項もありますので、Alphafold3の実行環境構築前にLicence and Disclaimerをご確認の上、ライセンスを遵守いただきますようお願いします。

データベースは月1回程度の頻度で更新を予定しています。上記の指定を行うことで、常にTSUBAME4.0内の最新データベースが指定されます。
特定のタイミングのデータベースで固定したい場合、module load 時にバージョン指定をしてください。

[バージョン確認手順]

```
[login/rNnN]$ module load alphafold3_database
[login/rNnN]$ module list
Currently Loaded Modulefiles:
 1) alphafold3_database/202411
```

7.7.3. LocalColabfold用データベース

LocalColabfoldで利用可能なデータベースです。

```
[rNnN]$ module load colabfold_database
```

を実行することにより、環境変数**COLABFOLD_DATA_DIR**にデータベースへのパスが格納されます。
LocalColabfoldの実行環境はご自身で用意いただく必要があります。
LocalColabfoldについては、東京科学大学 総合研究院 難治疾患研究所 森脇由隆先生が執筆されたQiitaを参照してください。

LocalColabfoldの利用にあたっては、Colabfoldのライセンスが適用されます。
LocalColabfoldの実行環境構築前にColabfoldのライセンスをご確認の上、ライセンスを遵守いただきますようお願いします。

データベースは月1回程度の頻度で更新を予定しています。上記の指定を行うことで、常にTSUBAME4.0内の最新データベースが指定されます。
特定のタイミングのデータベースで固定したい場合、module load 時にバージョン指定をしてください。

[バージョン確認手順]

```
[login/rNnN]$ module load colabfold_database
[login/rNnN]$ module list
Currently Loaded Modulefiles:
 1) colabfold_database/202411
```

付録. 新旧TSUBAME比較

付録.1. システム概要比較

T3/T4のシステム概要比較

	TSUBAME3.0	TSUBAME4.0
倍精度総理論演算性能	12.5PFLOPS	66.8PFLOPS
半精度総理論演算性能	47.2PFLOPS	952PFLOPS
総主記憶容量	135TiB	180TiB
総ハードディスク容量	15.9PB	44.2PB
総SSD容量		327TB
計算ノード数	540台	240台
総コア数	15,120コア	46,080コア
総GPU数	2,160台	960台
インターコネクト	Intel Omni-Path HFI 100Gbps	InfiniBand NDR200 200Gbps
インターネット接続	SINET5 100Gbps	SINET6 100Gbps

付録.2. 計算ノード比較

T3/T4の計算ノード比較

	TSUBAME3.0	TSUBAME4.0
演算部名	計算ノード HPE SGI ICE-XA 540台	計算ノード HPE Cray XD665 240台
ノード構成　　台あたり		
CPU	Intel Xeon E5-2680 v4 2.4GHz × 2 Socket	AMD EPYC 9654 2.4GHz × 2 Socket
コア数/スレッド	14コア / 28スレッド×2CPU	96コア / 192スレッド×2CPU
メモリ	256GiB	768GiB (DDR5-4800)
GPU	NVIDIA TESLA P100 for NVlink-Optimized Servers ×4	NVIDIA H100 SXM5 94GB HBM2e × 4
SSD	2TB	1.92TB NVMe U.2 SSD
インターコネクト	Intel Omni-Path HFI 100Gbps ×4	InfiniBand NDR200 200Gbps × 4

付録.3. ソフトウェア比較

T3/T4のソフトウェア比較

付録.3.1 システムソフトウェア比較

T3/T4の比較

	TSUBAME3.0 (2023.4.6 現在)	**TSUBAME4.0**
OS	SUSE Linux Enterprise Server 12 SP5	RedHat Enterprise Linux Server 9.3
ジョブスケジューラ	Univa Grid Engine 8.6.11	Altair Grid Engine 2023.1.1
コンパイラ	GCC 4.8.5, 12.2.0 Intel Compiler 23.0.0 NVIDIA HPC SDK 22.2	GCC 11.4.1 Intel oneAPI compiler 2024.0 and MKL NVIDIA HPC SDK 24.1 AOCC 4.1.0
MPI	Intel MPI 21.8.0 SGI MPT 2.16 OpenMPI 3.1.4	Intel MPI 2021.11 OpeMPI 5.0.2
CUDAライブラリ	11.0.3 (12.1.0利用可)	12.3.2
CUDAドライバ	450.172.01	545.23.08
OmniPathドライバ(OPA)	10.10.3.1.1	
InfiniBandドライバ(OFED)		23.10-1.1.9

付録.3.2 商用アプリケーション比較

T3/T4の商用アプリケーション比較

ソフトウェア名	概要	TSUBAME3.0	TSUBAME4.0
ANSYS	解析ソフトウェア	○	○
ABAQUS	解析ソフトウェア	○	○
ABACUS CAE	解析ソフトウェア	○	○
MSC Nastran	解析ソフトウェア	○	
MSC Patran	解析ソフトウェア	○	
Gaussian	量子化学計算プログラム	○	○
GaussView	量子化学計算プログラム プリポストツール	○	○
AMBER	分子動力学計算プログラム	○	○
Materials Studio	化学シミュレーションソフトウェア	○	○
Discovery Studio	化学シミュレーションソフトウェア	○	○
Mathematica	数式処理ソフトウェア	○	○
Maple	数式処理ソフトウェア	○	
AVS/Express	可視化ソフトウェア	○	
AVS/Express PCE	可視化ソフトウェア	○	
LS-DYNA	解析ソフトウェア	○	○ ※ANSYSに含む
COMSOL	解析ソフトウェア	○	○
Schrodinger	化学シミュレーションソフトウェア	○	○
MATLAB	数値計算ソフトウェア	○	○
VASP	量子分子動力学計算プログラム	○	○
Linaro forge(旧:Arm Forge)	デバッガ	○	○
Intel Compiler	コンパイラ	○	○ ※oneAPIに含む
PGI Compiler	コンパイラ	○	○ ※NVIDIA HPC SDKに含む

付録.3.3 フリーウェア比較

T3/T4のフリーウェア比較

ソフトウェア名	概要	TSUBAME3.0	TSUBAME4.0
GAMESS	ソルバ・シミュレータ	○	○
Tinker	ソルバ・シミュレータ	○	○
GROMACS	ソルバ・シミュレータ	○	○
LAMMPS	ソルバ・シミュレータ	○	○
NAMMD	ソルバ・シミュレータ	○	○
QUANTUM ESPRESSO	ソルバ・シミュレータ	○	○
CP2K	ソルバ・シミュレータ	○	○
OpenFOAM	ソルバ・シミュレータ、可視化	○	○
CuDNN	GPUライブラリ	○	○
NCCL	GPUライブラリ	○	○
Caffe	DeepLearningフレームワーク	○	
Chainer	DeepLearningフレームワーク	○	
TensorFlow	DeepLearningフレームワーク	○	○
DeePMD-kit	MD用DeepLearningフレームワーク	○	○
R	インタプリタ(Rmpi,rpudに対応)	○	○
clang	コンパイラ	○	○ ※AOCC
Apache Hadoop	分散データ処理ツール	○	○
POV-Ray	可視化	○	○
ParaView	可視化	○	○
VisIt	可視化	○	○
turbovnc	リモートGUI(X11) 表示	○	○
gnuplot	データ可視化	○	○
Tgif	画像表示・編集	○	○
GIMP	画像表示・編集	○	○
ImageMagick	画像表示・編集	○	○
TeX Live	TeX ディストリビューション	○	○
Java SDK	開発環境	○	○
PETSc	リニアシステムソルバ、ライブラリ	○	○
FFTW	高速フーリエ変換ライブラリ	○	○
DMTCP	チェックポイント・リスタート	○	○
Singularity	Linux container for HPC	○	○ ※Apptainer
Open OnDemand	HPC向けWebポータル		○

付録.4. ストレージ比較

T3/T4のストレージ比較

TSUBAME3.0	用途	マウント	容量	ファイルシステム
	Homeディレクトリ 共有アプリケーション配備	/home /apps	40TB	GPFS+cNFS
	高速ストレージ領域1	/gs/hs0	4.8PB	Lustre
	高速ストレージ領域2	/gs/hs1	4.8PB	Lustre
	高速ストレージ領域3	/gs/hs2	4.8PB	Lustre
	ローカルスクラッチ領域	/scr	各ノード1.9TB	xfs (SSD)
	共有スクラッチ領域	/beeond	計算ノード依存	BeeGFS

TSUBAME4.0	用途	マウント	容量	ファイルシステム
	高速ストレージ領域 Homeディレクトリ	/gs/fs /home	372TB	Lustre
	大容量ストレージ領域 共有アプリケーション配備	/gs/bs /apps	44.2PB	Lustre
	ローカルスクラッチ領域	/local	各ノード1.62TiB	xfs　SSD)