スーパーコンピュータシステムの使い方

京都大学学術情報メディアセンター

このサイトでは、京都大学 学術情報メディアセンターのスーパーコンピュータシステムの使い方を掲載しています。

ご不明な点やご質問がありましたら、問い合わせフォームより、ご連絡をお願いいたします。

システムのメンテナンス情報、障害情報等は、情報環境機構のお知らせをご参照ください。 また、ノードの故障については、障害情報をご参照ください。

目次

クイックスタート

ログインアカウントの有効化

初めてシステムを利用する方は、利用申請手続きの完了後に、利用者ポータルで利用開始手続きの実施が必要です。

詳しくは利用開始手続きをご覧ください。

システムへの接続方法

スーパーコンピュータへのログインは、SSH (Secure SHell) の鍵認証に限定しています。

- 接続方法：SSHの公開鍵認証
- SSH公開鍵：利用者ポータルから登録してください。前システムで鍵を登録されていた方は、引き継がれていますので登録作業は必要ありません。
- 接続先：
  - システムA：camphor.kudpc.kyoto-u.ac.jp
  - システムB/クラウド：laurel.kudpc.kyoto-u.ac.jp
  - システムC：cinnamon.kudpc.kyoto-u.ac.jp
  - システムG：gardenia.kudpc.kyoto-u.ac.jp
  - ファイル転送サーバ：hpcfs.kudpc.kyoto-u.ac.jp

- 秘密鍵には必ずパスフレーズを付けてください。ログインノード上にパスフレーズ未設定の秘密鍵が置かれていた場合は自動で削除されます。
- 複数人でのアカウント（利用者番号）の使いまわしは厳禁です。

詳しくはシステムへの接続方法をご覧ください。

ログイン環境

ログインノードごとに、自動でロードされるモジュール環境が異なります。 それぞれ次の表のバッチ処理の環境と、コンパイラの環境がロードされます。 どのログインノードからもシステム環境を切り替えることで、相互にバッチジョブの投入も可能です。

なお、下記表は2023年度の情報となります。

ログインノード	システム環境	バッチ処理環境	コンパイル環境
camphor.kudpc.kyoto-u.ac.jp	SysA	slurm	intel, intelmpi, PrgEnvIntel
laurel.kudpc.kyoto-u.ac.jp	SysB	slurm	intel, intelmpi, PrgEnvIntel
cinnamon.kudpc.kyoto-u.ac.jp	SysC	slurm	intel, intelmpi, PrgEnvIntel
gardenia.kudpc.kyoto-u.ac.jp	SysG	slurm	nvhpc, openmpi, PrgEnvNvidia

クラウドシステムの環境に切り替えたい場合は次のコマンドで切り替えることができます。

```
$ module switch SysCL
```

moduleコマンドの詳細は Modules をご覧ください。

ストレージの利用

データの保存のために、全てのユーザがホームディレクトリを利用できます。 パーソナルコース、グループコース、専用クラスタコースをお申し込みの場合は、大容量ストレージを利用することができます。 いずれの保存領域も、全てのログインノードおよび計算ノードから同じPATHでアクセスすることが可能です。

- ホームディレクトリ (/home)：100G
- 大容量ストレージ (/LARGE0, /LARGE1)：数TB〜数百TB （申請資源量に応じて設定）
- 高速ストレージ (/FAST)：数百GB〜数十TB （申請資源量に応じて設定、2024年度400GB〜1000GB分お試し提供）

詳しくはストレージの利用をご覧ください。

プログラムのコンパイル

クラウドシステムでは、Intelコンパイラがデフォルトで設定されています。

詳しくはコンパイラ・ライブラリをご覧ください。

プログラムの実行

Slurm ジョブスケジューラを用いたプログラム実行環境を提供しています。

詳しくはプログラムの実行をご覧ください。

利用可能なソフトウェア

ソフトウェア / ライブラリ に一覧を掲載しています。

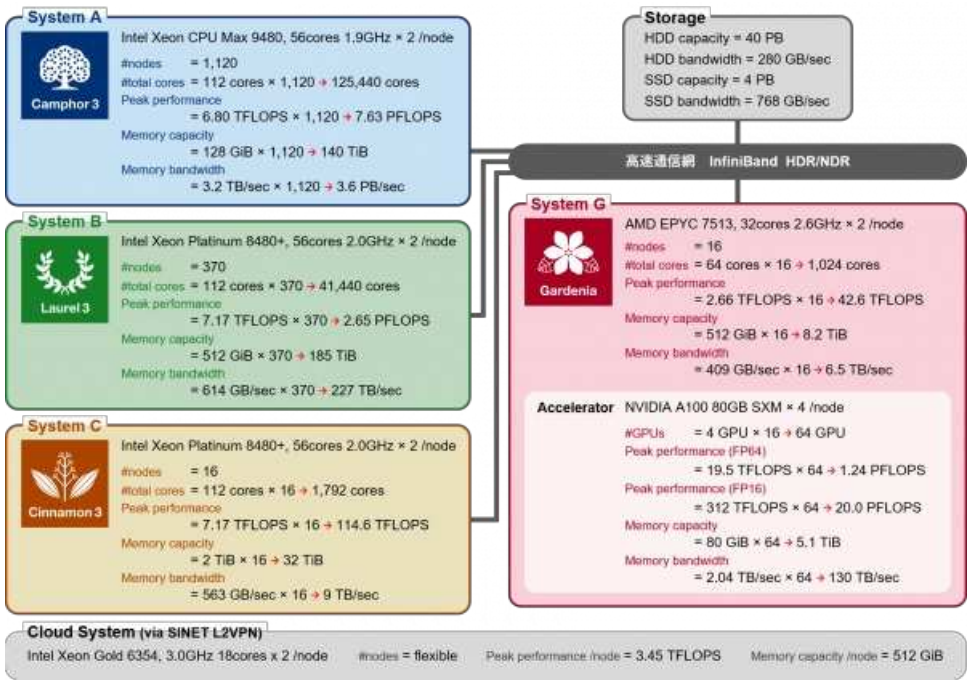
問い合わせ先

ご不明な点やご質問がありましたら、問い合わせフォームより、ご連絡をお願いします。

システムへの接続方法

システム構成

京大スパコンシステムは、5種のシステムとストレージにより構成されます. 詳細は情報環境機構HPをご覧ください。



ログインまでの流れ

スーパーコンピュータへのログインは、SSH (Secure SHell) の鍵認証に限定しています。 鍵ペアを作成し、 利用者ポータルから公開鍵の登録が完了するとログインできます。 パスワードでのログインはできませんので、必ず最初に公開鍵の登録が必要です。

ログインまでの流れは下図のようになります。具体的な手順は、 システムへの接続方法をご覧ください。

ホスト名(FQDN)

システムごとのホスト名は以下の通りです。利用承認書に記載のログイン名 (利用者番号) と利用者ポータルで登録した公開鍵に対応する秘密鍵を使用して接続してください。

システム名	ホスト名	備考

システム名	ホスト名	備考
システムA	camphor.kudpc.kyoto-u.ac.jp	2台のログインノードで構成。
システムB/クラウド	laurel.kudpc.kyoto-u.ac.jp	3台のログインノードで構成。
システムC	cinnamon.kudpc.kyoto-u.ac.jp	3台のログインノードで構成。
システムG	gardenia.kudpc.kyoto-u.ac.jp	2台のログインノードで構成。
ファイル転送サーバ	hpcfs.kudpc.kyoto-u.ac.jp	2台のサーバで構成。時間制限のないSFTP、RSYNC専用のサーバ。

fingerprint

システムごとのfingerprintは以下の通りです。システムに初めてSSH接続する際に表示されるfingerprintの確認にお使いください。

システム名	方式	**fingerprint**
システムA	RSA	**SHA256:** o/Ef0rC2uksvUax14XF6R9c3WHWypaSDfDjDJ0lkreQ
"	ED25519	**SHA256:** yoBITHVW+ENaAAAIW+ZZDCUBnCNFEhBDsnWHjMHFKx0
"	ECDSA	**SHA256:** 8j+LvvPha40b2ZwrN3J7s3fyzD+SxHU67/5MwKAUqVU
システムB/システムC/クラウド	RSA	**SHA256:** c3KPtVlPbgGW+jFM8VdGd4yob4HX/VPH7nR26JHS13M
"	ED25519	**SHA256:** jt5nEfylQ+SjG5PUZtKqP1DXk56p+7ugwF8GHx8Nr/Q
"	ECDSA	**SHA256:** yJN7gkHzsrlRBwvYsTCiQkXnzEkdZmjMZLs9TO8YylU
システムG	RSA	**SHA256:** 7GWj0iRCCMpswAWWylZC+FMX3xlQVRyFiP9hIUYl3rU
"	ED25519	**SHA256:** 8uh/6B14HEz77F1yAq95CyQHLqKqr4DG2Xys7eQr0qk
"	ECDSA	**SHA256:** PJX5f4jPqjTM4Awpod8fedTEXknIBamdexYK3fAzl6U
ファイル転送サーバ	RSA	**SHA256:** /ZeTSRgrxAo4Et1E9NouKs8xyhc/KAjgRpqaEgf37sM
"	ED25519	**SHA256:** v0ABGIXX735Ak2By6zq1luwPAPwLmJC8IwEhzAlm7ds
"	ECDSA	**SHA256:** RtD7xSYrSL/F6KZpZcjluDFlQ37CIVD6ij2GhEYOaO0

ログイン手順

GUIを必要としない場合、SSHクライアントを使用してログインします。
MobaXtermでログインすることも可能です。

X (GUI) を利用する場合のログイン手順

FastXや、NiceDCV、MobaXterm(Windowsのみ)を ご利用いただけます。

ファイル転送の手順

スーパーコンピュータとのファイル転送は、SSHの機能でファイル転送を行うscpまたはsftpを利用します。SCPは安全なコピー（Secure CoPy）、SFTPは安全なFTP（Secure FTP）です。ファイル転送を行う場合にも、 利用者ポータルで登録した公開鍵に対応する秘密鍵を使用して接続する必要があります。

SSHクライアントを使用してファイル転送を行う手順はSSHによるファイル転送をご覧ください。

ログインノードの一部障害時のログイン方法

スーパーコンピュータのログインノードは、システムB、C、クラウド共通で3台、システムGで2台あり、負荷分散のためにDNSラウンドロビンで運用しています。その内の一部のノードで障害が発生するとDNSラウンドロビンでのログインに失敗することがありますので、次の手順で個別のホスト名を指定してログインしてください。

システム名	ホスト名
システムA	`camphor31.kudpc.kyoto-u.ac.jp`
"	`camphor32.kudpc.kyoto-u.ac.jp`
システムB/クラウド	`laurel31.kudpc.kyoto-u.ac.jp`
"	`laurel32.kudpc.kyoto-u.ac.jp`
"	`laurel33.kudpc.kyoto-u.ac.jp`
システムC	`cinnamon32.kudpc.kyoto-u.ac.jp`
"	`cinnamon32.kudpc.kyoto-u.ac.jp`
"	`cinnamon33.kudpc.kyoto-u.ac.jp`
システムG	`gardenia11.kudpc.kyoto-u.ac.jp`
"	`gardenia12.kudpc.kyoto-u.ac.jp`

強力なパスワード・パスフレーズの作成について

スーパーコンピュータシステムの利用にあたり、利用者ポータルのパスワード、および鍵認証における秘密鍵のパスフレーズを利用者自身で厳重に管理していただく必要があります。また、パスワード・パスフレーズの作成にあたっては、アカウント情報等から第三者が容易に推察できないような文字列を設定してください。

安全性の高い、強力なパスワード・パスフレーズを作成するためのヒントとして、以下のサイトをご参考ください。

- 強力なパスワードを作成および使用する (Microsoft)
- パスワードチェッカー (Microsoft)

- Macで安全なパスワードを作成するためのヒント (Apple)

鍵ペアの作成と公開鍵の登録

鍵認証とは

鍵認証とは、パスワード認証と異なり、予め登録した公開鍵とPCに保存した秘密鍵のペアをログイン時の認証に使用する認証方法です。従来の、パスワードでの認証は、辞書攻撃などによりパスワードが悪意の第三者に使用されてしまう可能性がありますが、鍵認証では秘密鍵の管理さえしっかりしていれば、そのような心配はありません。 また、秘密鍵はパスフレーズ（長い文字列を使用したパスワード）による保護が可能であり、PCの紛失等で秘密鍵が漏洩した場合でも第三者による秘密鍵の悪用を防ぐことができます。

パスフレーズなし秘密鍵の利用は禁止していますので必ず設定してください。ログインノード上にパスフレーズなし秘密鍵が置かれている場合、自動で削除処理が行われます。

鍵ペアの作成

手順

Windowsの場合は「コマンドプロンプト」、Macの場合は「ターミナル」でssh-keygenコマンドを使用して鍵ペアを作成します。 Windowsでは、MobaXtermで鍵を生成することも可能です。

```
$ ssh-keygen -t rsa -b 3072 -m pem  #(1)
Generating public/private rsa key pair.
Enter file in which to save the key (/home/taro/.ssh/id_rsa): #(2)
Enter passphrase (empty for no passphrase):  #(3)
Enter same passphrase again:  #(4)
Your identification has been saved in /home/taro/.ssh/id_rsa.
Your public key has been saved in /home/taro/.ssh/id_rsa.pub.
The key fingerprint is:
8c:13:10:d2:c0:12:c5:0b:53:d4:3f:b6:9c:16:f6:ca taro@test.kyoto-u.ac.jp
```

作成した鍵ペアの確認

デフォルトの保存先で鍵ペアを作成した場合、 %HOMEPATH%.ssh (Windowsの場合)、~/.ssh (Mac/Linuxの場合)を確認すると以下のように鍵が作成されています。 **id_rsa** が秘密鍵、 **id_rsa.pub** が公開鍵です。スパコンへのログインの前に、 利用者ポータルでの公開鍵の登録 の手順で、公開鍵を登録していただく必要があります。

- Windowsの場合

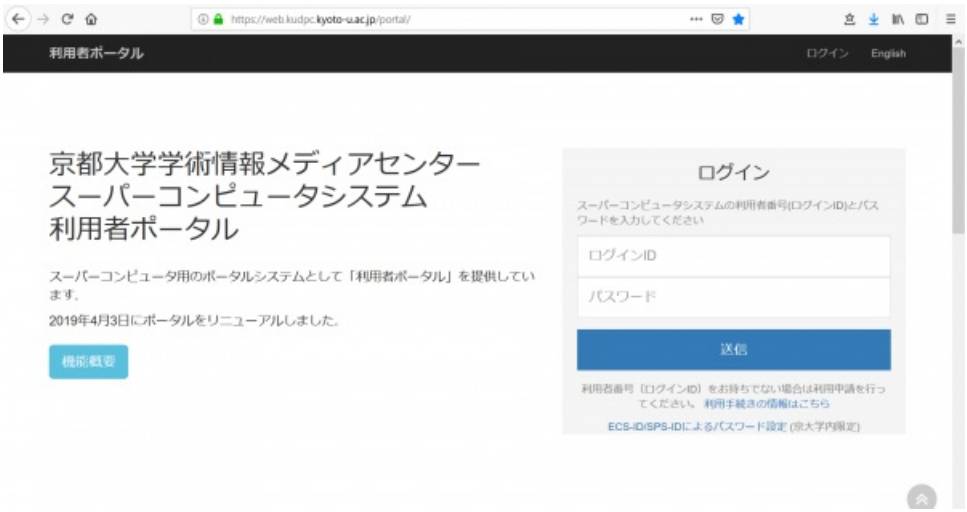
```
>dir %HOMEPATH%\.ssh
2022/10/01  01:23             1,675 id_rsa
2022/10/01  01:23               410 id_rsa.pub
>type %HOMEPATH%\.ssh\id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEAsotK4PbdadfhbXbTPIsxvwKFIg+8Lmp0pXKckAOuSnoaaT516ddj9rnIJlE/JaJf0cl
```

- Mac/Linuxの場合

```
$ ls ~/.ssh/
id_rsa id_rsa.pub
$ cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEAsotK4PbdadfhbXbTPIsxvwKFIg+8Lmp0pXKckAOuSnoaaT516ddj9rnIJlE/JaJf0cl
```

利用者ポータルでの公開鍵の登録

1. 利用者ポータル へアクセスし、利用者番号（ユーザ名）とパスワードを入力してください。



2. 左メニューの **SSH公開鍵** を選択してください。

3. **SSH**用公開鍵の追加で タイトル(任意)を入力し、 公開鍵 に作成した公開鍵をコピー＆ペーストし、 秘密鍵のパスフレーズ設定 を遵守していることを確認した上で、 追加 を押すと登録が完了します。

4. 登録の完了後、作成した秘密鍵とパスフレーズを使用して、スーパーコンピュータにログインできます。ログインの具体的な手順は、 こちら をご覧ください。

鍵の取り扱いについて

鍵の管理

秘密鍵が流出すると、スーパーコンピュータシステムのセキュリティに対して重大な脅威になります。秘密鍵が他人に漏れることのないよう適切に管理してください。なお、他人に漏れた恐れがある場合には、portal * kudpc.kyoto-u.ac.jp (*

は @ に変えてください) にご連絡いただき適切な処置についてご相談ください。

複数PCからスパコンにログインする場合の鍵の取扱い

デスクトップPCとノートPCなど、複数PCからスーパーコンピュータシステムにログインする場合など、複数PCからログインする場合、鍵ペアはそれぞれ別のものを使用してください。

公開鍵の追加、削除

利用者ポータルでの公開鍵の登録を参考に追加/削除を実施してください。

秘密鍵の紛失、パスフレーズ忘れなどの場合

利用者ポータルでの公開鍵登録は、スーパーコンピュータへのログイン成功後一定時間が経過すると停止します。秘密鍵の紛失、パスフレーズ忘れの場合は、登録いただいているメールアドレスから。以下の事項を記載の上、お申し出ください。

- 宛先：consult * kudpc.kyoto-u.ac.jp (* は @ に変えてください)
- 差出人：登録いただいているメールアドレス
- 件名：公開鍵登録の再開希望
- 本文：
  - 利用者番号：
  - 所属機関・部局：
  - 氏名：
  - 秘密鍵の紛失、パスフレーズ忘れの状況、理由

SSHでのログイン

前提条件

鍵ペアの作成と公開鍵の登録 の手順で鍵ペアを作成し、 利用者ポータルから公開鍵を登録していること。

接続方法

1. Windowsの場合は「コマンドプロンプト」、Macの場合は「ターミナル」を起動します。
   Windowsでは、MobaXtermで接続することも可能です。MobaXtermはx11サーバ、SSHクライアント、ファイル転送等を行うことができる便利なツールボックスです。
2. sshコマンドでログインノードに接続します。

```
$ ssh ユーザ名@ホスト名
```

3. [ユーザ名@ホスト名 ~] の形式のプロンプトが表示されればログイン成功です。

接続先

システムごとのホスト名は以下の通りです。

システム名	ホスト名	備考
システムA	camphor.kudpc.kyoto-u.ac.jp	2台のログインノードで構成。
システムB/クラウド	laurel.kudpc.kyoto-u.ac.jp	3台のログインノードで構成。
システムC	cinnamon.kudpc.kyoto-u.ac.jp	3台のログインノードで構成。
システムG	gardenia.kudpc.kyoto-u.ac.jp	2台のログインノードで構成。

実行例

- b59999の利用者番号でシステムAにログインする

```
$ ssh b59999@camphor.kudpc.kyoto-u.ac.jp
Enter passphrase for key 'id_rsa':
```

- 秘密鍵を指定して、b59999の利用者番号でシステムBにログインする

```
$ ssh -i ~/.ssh/id_rsa_kudpc b59999@laurel.kudpc.kyoto-u.ac.jp
Enter passphrase for key 'id_rsa':
```

FastXでのログイン

FastXとは

FastXは、HTTPS(443/TCP)を用いて、スーパーコンピュータシステムに 接続することが可能な X Window Systemです。 データ圧縮技術により通信トラフィックが抑えられており、GUIベースの アプリケーションの操作などを遠隔地からも快適に実行いただく事が可能です。

FastXによるリモートアクセスサービスをご利用いただく際には、Google ChromeやFirefoxなどのWebブラウザが必要です。 専用のクライアントソフトのインストールやWebブラウザに別途プラグインをインストール する必要ありません。

前提条件

- 鍵ペアの作成と公開鍵の登録 の手順で鍵ペアを作成し、 利用者ポータル から公開鍵を登録していること。また、秘密鍵は**RSA**の**OpenSSH**形式**(7.7**以前の**PEM**形式**)**でも保存してあること。
- 動作確認の取れているブラウザ：Edge, Firefox, Chrome
    - 双方向のコピーペーストはChromeのみサポートします。他のブラウザの場合、サーバ側からユーザ側への片方向のみ有効です。

秘密鍵のフォーマットについて

FastXはECDSA形式およびED25519形式の鍵が利用できませんので、RSA形式の鍵をご利用ください。 また、OpenSSH 7.8 以降のフォーマットの秘密鍵が読み込めません。 OpenSSH 7.8 以降のフォーマットでは、秘密鍵は下記の内容となります。

```
-----BEGIN OPENSSH PRIVATE KEY-----
**************************************
**************************************
**************************************
-----END OPENSSH PRIVATE KEY-----
```

ssh-keygen コマンドで OpenSSH 7.8 以前のフォーマットで秘密鍵を作成するには、下記のように `-m pem` オプションをつけてください。

```
$ ssh-keygen -t rsa -b 3072 -m pem
```

OpenSSH 7.8 以前のフォーマット（PEM形式）の秘密鍵は、以下の内容となります。

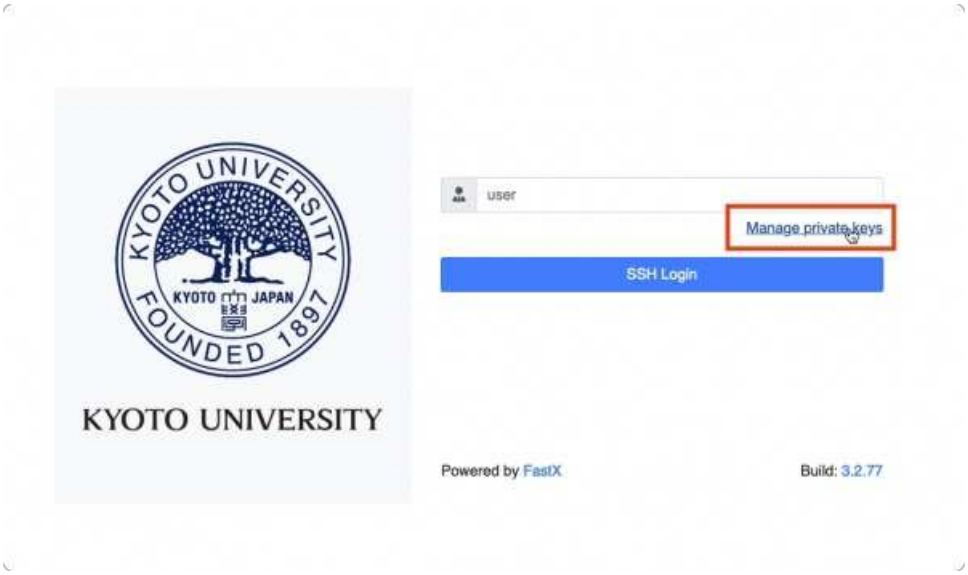
```
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: AES-128-CBC,7A30F993641D4093A373703A3F644D2D

**********************************
**********************************
**********************************
-----END RSA PRIVATE KEY-----
```

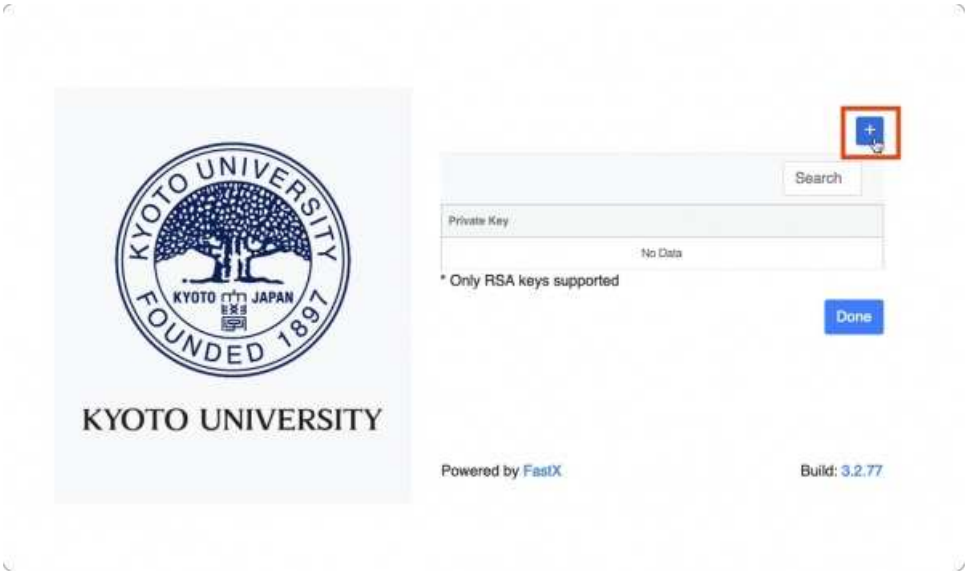
接続手順

1. ブラウザを起動し、以下のいずれかのアドレスに移動します。

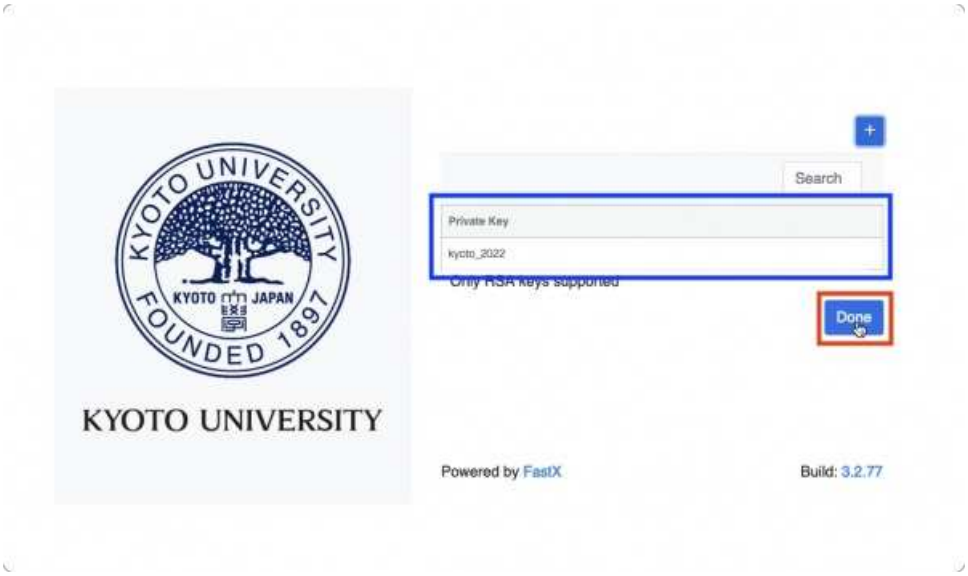
接続先	アドレス
システムA	https://camphor31.kudpc.kyoto-u.ac.jp/
システムB/C	https://laurel31.kudpc.kyoto-u.ac.jp/
システムG	https://gardenia31.kudpc.kyoto-u.ac.jp/
アプリケーションサーバ	https://app.kudpc.kyoto-u.ac.jp/

2. （初回のみ）「Manage Private Keys」をクリックします。秘密鍵のアップロードを促す小窓が開きますので、「＋」をクリックし、**OpenSSH**形式の秘密鍵を選択します。

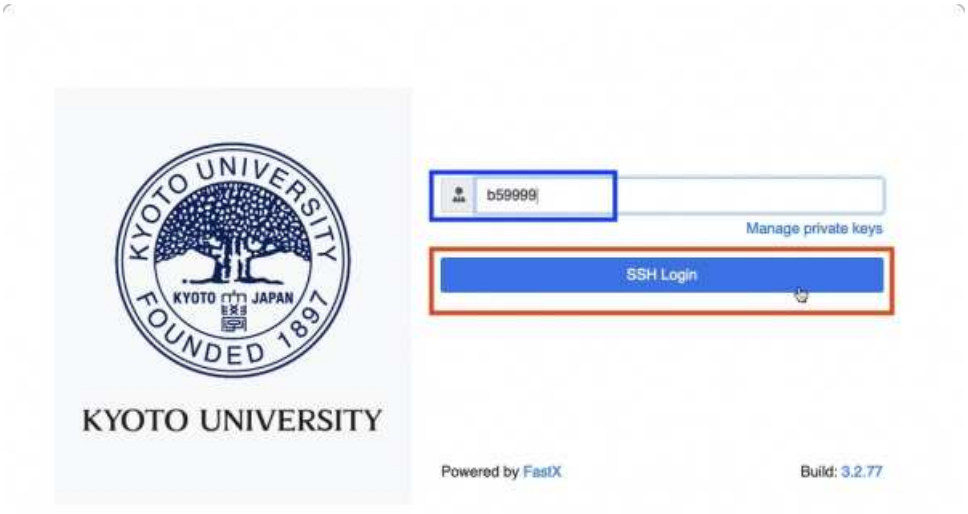




3. （初回のみ）青色の枠の部分にアップロードした秘密鍵の名前が表示されていることを確認して，「Done」をクリックします．

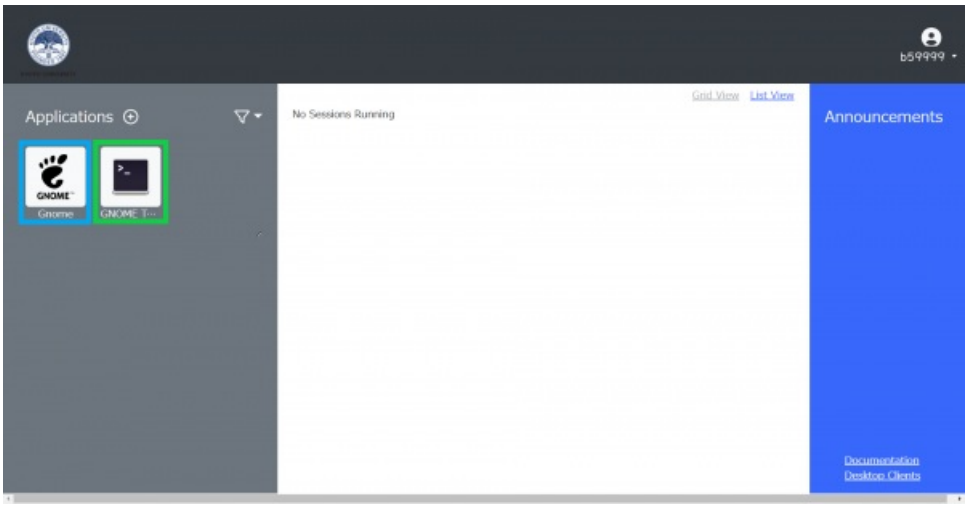


4. 青色の枠の部分に利用者番号を入力して「SSH Login」をクリックします。

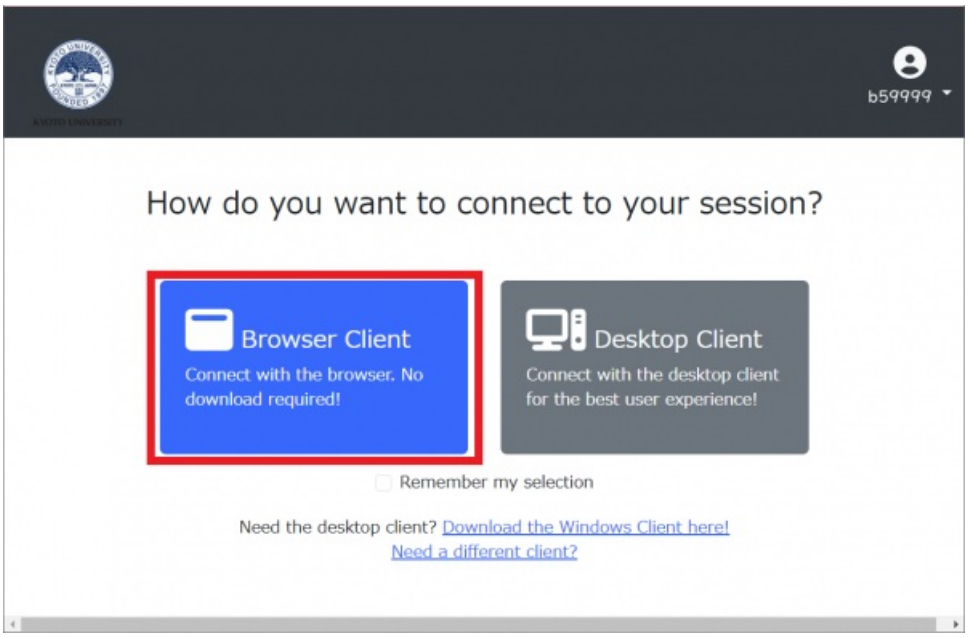


5. 青色の枠の部分にパスフレーズを入力して「Submit」をクリックします。

6. 起動するアプリケーションを選択します。
   6-1. ターミナルを使う場合は、緑色の枠に囲まれた「GNOME Terminal」をダブルクリックします。
   6-2. デスクトップ画面を使う場合は、青色の枠に囲まれた「GNOME」をダブルクリックします。

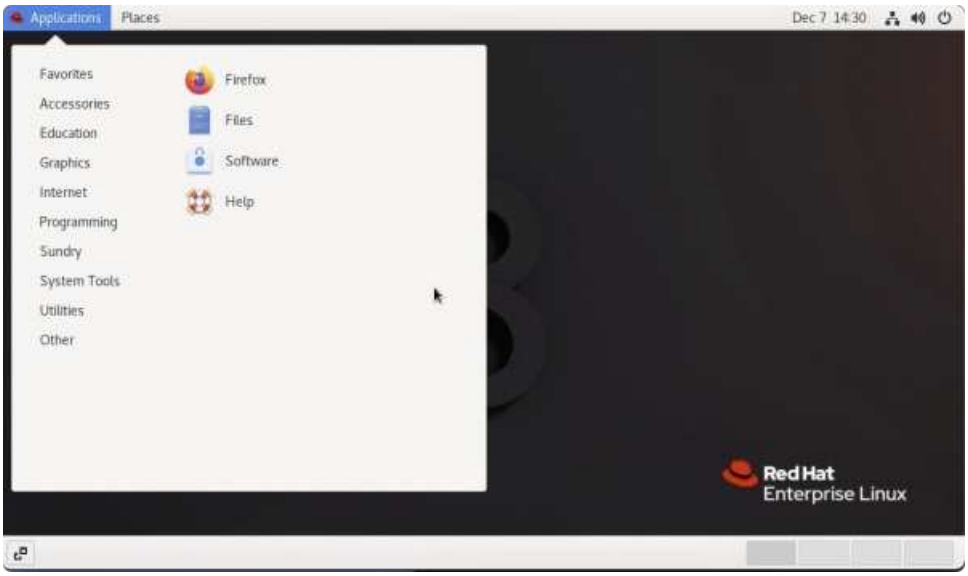


7. Browser Clientをクリックします。

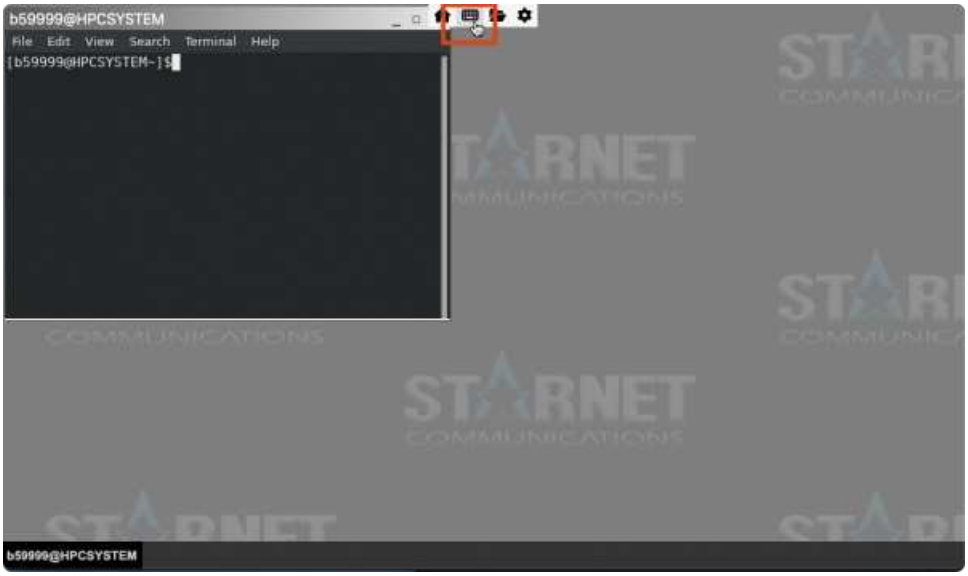


8. 画面が表示されればログイン成功です。
   手順6 で 6-1. ターミナル「GNOME Terminal」を選択した場合

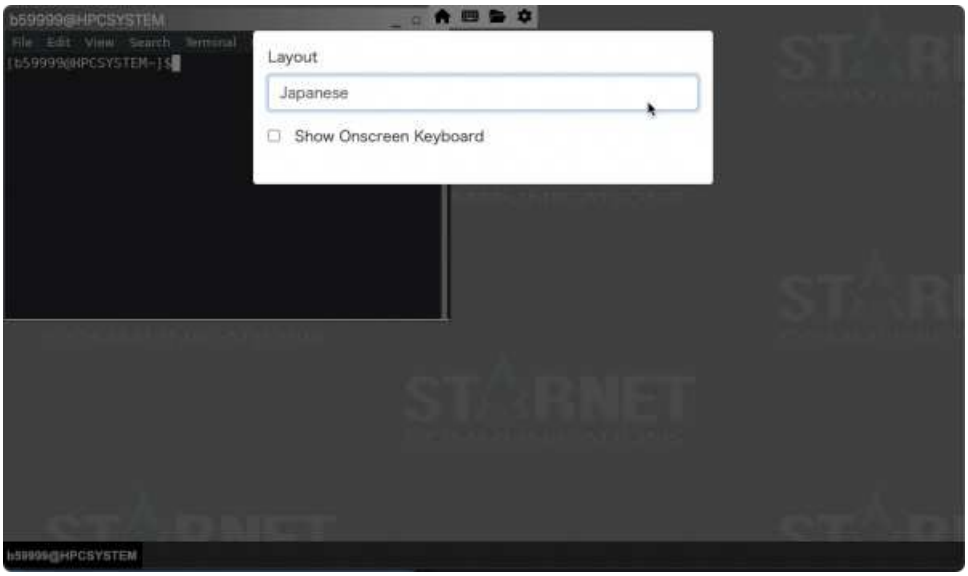
   手順6 で 6-2. デスクトップ画面「GNOME」を選択した場合
   GNOME のスクリーンロック時のパスワードは、利用者ポータルのパスワードです（秘密鍵のパスフレーズではありません）



9. デフォルトではキーボードレイアウトは「Japanese」に設定しております。他言語に設定する場合は、画面上のタブを引き出し、キーボードマークをクリックします。



10. Layoutを所望の言語に変更ください。



注意事項

同時接続数について

一人のユーザが同時に複数のセッションを利用する（デスクトップやターミナルを開く）場合、同じログインノード上ではライセンスは1つしか消費しませんが、別々のログインノードからセッションを開くと、その分のライセンスが消費されます。なお、1ユーザの利用可能なライセンス数は3に設定しております。それ以上のセッションを開こうとした場合、ライセンス不足の旨のnoticeが表示されます。

セッションのタイムアウトについて

FastXでは、切断してもセッションは維持され、ユーザによる再接続が可能ですが、切断後に再接続しないまま2日経過すると、セッションが自動で削除されますのでご注意ください。

Anaconda使用時の接続について

`$ conda init <シェル名>` を実行すると、Anaconda設定用のコマンドが$HOME/.bashrcに追加されます。その状態で、FastXのGNOME Terminalで接続しようとすると "Client Disconnected" と表示され、接続できない事象を確認しています。その場合は、$HOME/.bashrcに追加されたコマンドの、`. "/opt/system/app/anaconda3/2022.10/etc/profile.d/conda.sh"` （anaconda3/2022.10の場合）を残して、それ以外の行を削除してください。anaconda3/2022.10以外の場合は、適宜読み替えてください。

```
. "/opt/system/app/anaconda3/2022.10/etc/profile.d/conda.sh"
```

参考情報

クライアントソフトウェアについて

クライアントソフトは、開発元の公式Webサイトにて配布されています。 必要に応じてご利用ください。

クライアントから接続する際の通信プロトコルは、**SSH** としてください。 また、パスワード認証はご利用いただけませ

んので、SSHエージェント(Windowsの場合は、Puttyに内包されている Pageant、macOSやLinuxはssh-agentなど)が、 必要です。

Nice DCVでのログイン

Nice DCVとは

Nice DCVは、サーバ側のGPUを利用し、可視化アプリケーション等の2D/3Dグラフィックを高速に描画可能なリモートデスクトップソフトウェアです。 データ圧縮技術により通信トラフィックを抑えているため、遠隔地でも快適にGUIベースのアプリケーションをご利用いただけます。 専用のクライアントソフト（Windows, Mac, Linux)からご利用頂けます。

前提条件

- 下記のサイトからクライアントソフトをダウンロードし、インストールして下さい。
  https://download.nice-dcv.com/
- 鍵ペアの作成と公開鍵の登録 の手順で鍵ペアを作成し、利用者ポータル から公開鍵を登録していること。
- SSHでのログイン で、アプリケーションサーバへのログインができること。
  - Windowsでは、MobaXtermで接続することも可能です。MobaXtermはx11サーバ、SSHクライアント、ファイル転送等を行うことができる便利なツールボックスです。

接続手順

NiceDCVは、以下のアプリケーションサーバでご利用いただくことが可能です。

接続先	アドレス
アプリケーションサーバ	app.kudpc.kyoto-u.ac.jp

NiceDCVは、外部から直接利用いただくことができないため、SSHのポートフォワーディング機能を用いて アプリケーションサーバに接続いただく必要があります。 以下の何れかの手順を参考に接続をお願いします。

ポートフォワーディングの設定

コマンドプロンプトやターミナルなどを用いる場合

1. sshコマンドでアプリケーションサーバへログインします。その際、**localhost:5901** へのポートフォワーディングを行います。

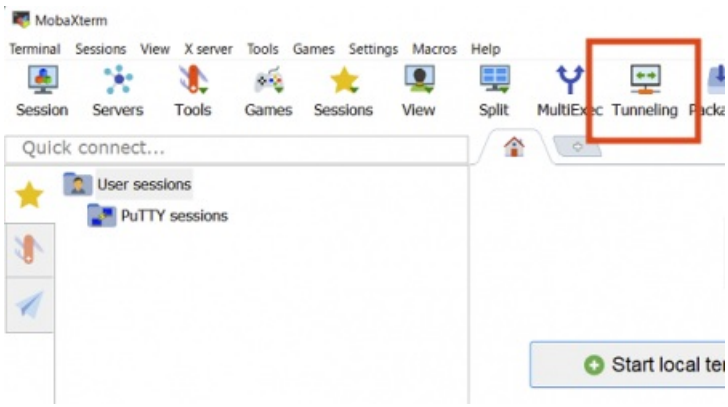
```
$ ssh {利用者番号}@app.kudpc.kyoto-u.ac.jp -L 5901:localhost:5901

例：b59999 ユーザでログインする場合
$ ssh b59999@app.kudpc.kyoto-u.ac.jp -L 5901:localhost:5901
```

2. コマンドプロンプトやターミナルを閉じずに、Nice DCVクライアントの設定 へお進み下さい。

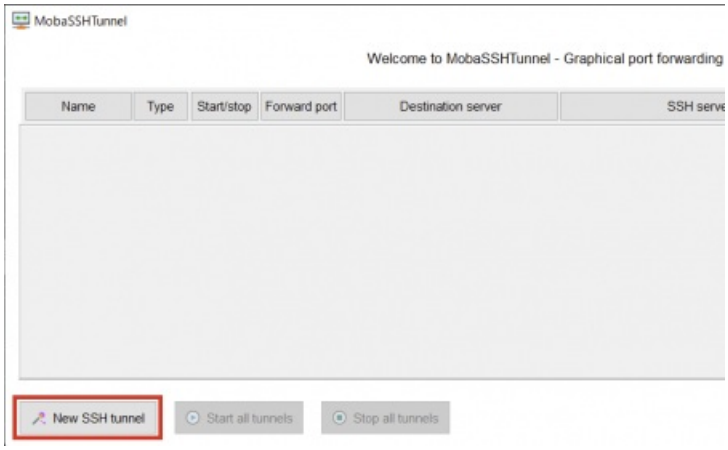
MobaXtermを用いる場合(Windowsのみ)

1. MobaXtermの利用方法を参考に、MobaXtermを起動します。

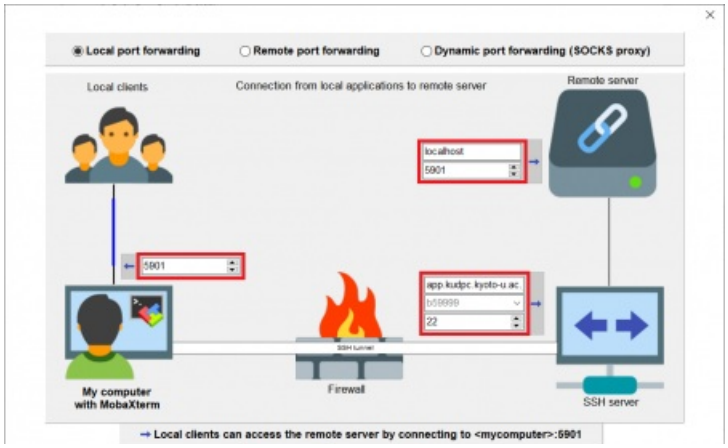
2. ツールバーから「Tunneling」をクリックします。



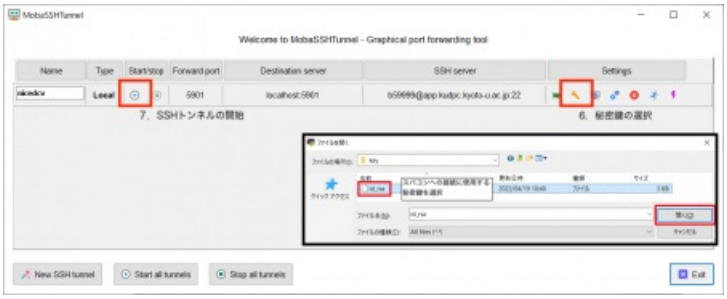
3. 「New SSH tunnnel」をクリックします。



4. 「Local port forwarding」にチェックを入れ、下記の値を入力します。
    - <Forwarded port>：5901
    - <SSH server>：app.kudpc.kyoto-u.ac.jp
    - <SSH login>：スーパーコンピュータシステムの利用者番号(ID)
    - <SSH port>：22
    - <Remote server>：localhost
    - <Remote port>：5901
5. 「Save」をクリックします。



6. 鍵のマークをクリックし、鍵ペアの作成と公開鍵の登録で作成した秘密鍵を選択します。
7. Startボタンをクリックします。



8. アプリケーションサーバへログインします。
9. **MobaXterm**の画面を閉じずに、Nice DCVクライアントの設定 へお進み下さい。

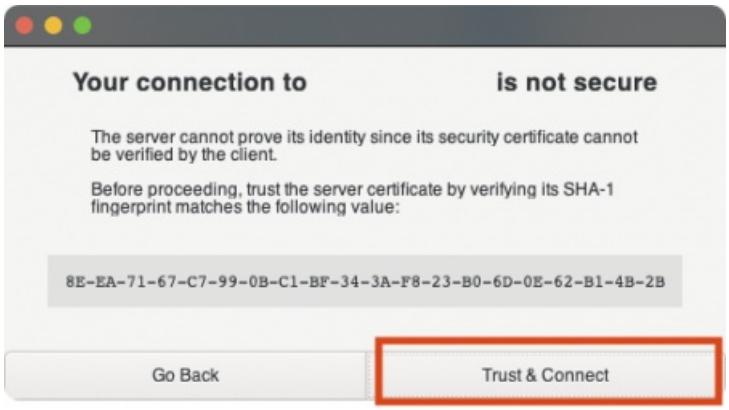
Nice DCVクライアントの設定

1. ポートフォワーディングの設定 で接続したコンソール上で、以下のコマンドを実行し セッションの登録を行って下さい。

```
$ dcv create-session {利用者番号}

(例)
$ dcv create-session b59999
```

2. Nice DCVのクライアントソフトを起動後、入力欄に **localhost:5901#**利用者番号 と入力し、Connect をクリックします。

3. 初回接続時は、ポートフォワーディングで接続いただく都合上、証明書のドメイン名と接続時のドメイン名(localhost)に 差分があるため，以下のような警告画面が表示されます。 表示されるfingerprintが、以下の何れかに一致している場合は、「Trust&Connect」を押下して、進んでください。



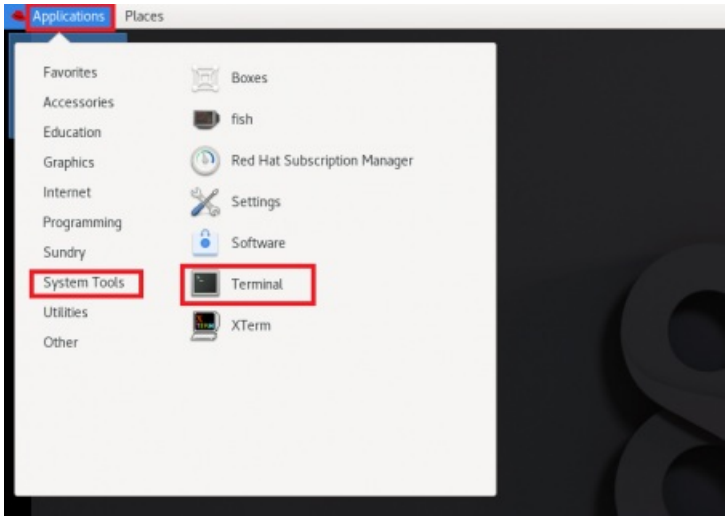
> app.kudpc.kyoto-u.ac.jp で使用する証明書のfingerprint

fingerprint
8E-EA-71-67-C7-99-0B-C1-BF-34-3A-F8-23-B0-6D-0E-62-B1-4B-2B

3. "UserName" に利用者番号、"Password" に「利用者ポータル」にログインする際に使用するパスワードを入力し、Login をクリックします。

4. 接続に成功すると、GNOME Desktopが表示されます。なお、GNOME Desktop のスクリーンロック時のパスワードは、「利用者ポータル」にログインする際に使用するパスワードです（秘密鍵のパスフレーズではありません）

5. 「Applications」、「System Tools」、「Terminal」の順にクリックし、Terminalを起動します。



利用終了手順

1. ポートフォワーディングの設定 で接続したコンソール上で、以下のコマンドを実行し セッションの終了処理を行って下さい.

```
$ dcv close-session {利用者番号}

(例)
$ dcv close-session b59999
```

補足事項

同時セッション数について

1ユーザが同時に登録できるセッションは1つまでとさせて頂きます。 なお、同一のセッションへの同時接続には制限はありません。

資源の制限

- アプリケーションサーバは、すべてのユーザがノードを共有して利用する形態となっております。
- 1ユーザが同時に利用できるCPUコア数は、**8コア** となっており変更することはできません。
- 1ユーザが同時に利用できるメモリ量は、**64GB** となっており変更することはできません。
- セッションを登録してからの経過時間が24時間を超えると、セッションは強制終了されます。 終了時にセッションはリセットされ、レジュームは行われませんので、ご注意ください。

dcvコマンドについて

dcv コマンドは、セッションの登録、詳細の確認、終了に対応しています。 「b59999」の部分は、ご自身の大型計算機システム利用者番号に置き換えてご利用ください。

- セッションの登録

```
$ dcv create-session b59999
```

- セッションの詳細の確認

```
$ dcv describe-session b59999
  Session:
id: b59999
owner: b59999
X display: :1
X authority: /run/user/59999/dcv/b59999.xauth
display layout: 800x600+0+0
```

- セッションの終了

```
$ dcv close-session b59999
```

SSHによるファイル転送

前提条件

鍵ペアの作成と公開鍵の登録の手順で鍵ペアを作成し、利用者ポータルから公開鍵を登録していること。

ファイルの転送方法

scp コマンドおよび sftp コマンドを利用して接続する手順を紹介します。Windowsでは、MobaXtermで接続することも可能です。MobaXtermでは、GUIでファイル転送が可能です。

接続先

システムごとのホスト名は以下の通りです。

システム名	ホスト名	備考
ファイル転送サーバ	hpcfs.kudpc.kyoto-u.ac.jp	推奨 2台のサーバで構成。時間制限のないSFTP、RSYNC専用のサーバ。
システムA	camphor.kudpc.kyoto-u.ac.jp	2台のログインノードで構成。
システムB/クラウド	laurel.kudpc.kyoto-u.ac.jp	3台のログインノードで構成。
システムC	cinnamon.kudpc.kyoto-u.ac.jp	3台のログインノードで構成。
システムG	gardenia.kudpc.kyoto-u.ac.jp	2台のログインノードで構成。

scpコマンドによるファイル転送

```
$ scp［オプション］［転送元のパス］［転送先のパス］
```

使用例

リモートマシンはシステムB、利用者番号はb59999とします。

- ローカルマシンのfile1.txtをリモートマシンのホームディレクトリに転送する

```
$ scp file1.txt b59999@laurel.kudpc.kyoto-u.ac.jp:
Enter passphrase for key 'id_rsa':
```

- リモートマシンのfile1.txtをfile2.txtとしてローカルマシンに転送する

```
$ scp b59999@laurel.kudpc.kyoto-u.ac.jp:file1.txt file2.txt
Enter passphrase for key 'id_rsa':
```

- ローカルマシンのディレクトリdir1をリモートマシンに転送する

```
$ scp -r dir1 b59999@laurel.kudpc.kyoto-u.ac.jp:
Enter passphrase for key 'id_rsa':
```

- 秘密鍵を指定したうえで、ローカルマシンのfile1.txtをリモートマシンに転送する

```
$ scp -i id_rsa file1.txt b59999@laurel.kudpc.kyoto-u.ac.jp:
Enter passphrase for key 'id_rsa':
```

sftpコマンドによるファイル転送

```
$ sftp [オプション] [接続先]
```

使用例

リモートマシンはシステムB、利用者番号はb59999とします。

```
$ sftp b59999@laurel.kudpc.kyoto-u.ac.jp
Enter passphrase for key 'id_rsa':
Connected to laurel.kudpc.kyoto-u.ac.jp.
sftp> put file1.txt
Uploading file1.txt to /home/b/b59999/file1.txt
file1.txt                                                                   100%   31     9.8
sftp> ls
file1.txt
sftp> get file2.txt
Fetching /home/b/b59999/file2.txt to file2.txt
/home/b/b59999/file2.txt                                                  100%   31     4.2KB/
sftp> quit
```

Archaeaによるファイル転送

Archaea(旧HCP Tools)とは

高速なファイル転送ツールとして、Archaea(旧HCP Tools)を提供しています。

Archaea(旧HCP Tools)は、HpFPというプロトコルを使用しているため、高速なファイル転送が可能です。専用のパッケージをインストールしていただくことでご利用いただけます。

前提条件

- Archaea(HCP Tools)クライアントのインストールの手順でArchaea(旧HCP Tools)クライアントのインストールが完了していること。
- 鍵ペアの作成と公開鍵の手順で鍵ペアを作成し、利用者ポータルから公開鍵を登録していること。
- 上記手順で作成した、**RSA**形式 の秘密鍵を以下のパスに配置していること。

```
・Windowsをクライアントとして利用する場合：
    ~/_hcp/id_rsa
・Linuxをクライアントとして利用する場合：
    ~/.hcp/id_rsa
```

秘密鍵の保管場所は設定ファイルで変更可能です。詳しい設定方法はオンラインドキュメントをご覧ください。

ファイル転送の方法

接続先

システム先	ホスト名	備考
ファイル転送サーバ	hpcfs.kudpc.kyoto-u.ac.jp	2台のサーバで構成。

接続先にはファイル転送サーバ(hpcfs.kudpc.kyoto-u.ac.jp)を指定してください。システムB(laurel.kudpc.kyoto-u.ac.jp)等には接続できません。

Archaea toolsによるファイル転送

hcpコマンドを利用して接続する手順を紹介します。 hcpコマンドはコマンドプロンプト(ターミナル)を使用します。

使い方

```
アップロード : hcp [OPTION]... SOURCE [SOURCE]... [USER@]HOST[:PORT]:DEST
ダウンロード : hcp [OPTION]... [USER@]HOST[:PORT]:SOURCE [:SOURCE]... DEST
```

オプション

ショートオプション	ロングオプション	説明
-p	--permission	転送元のパーミッションを保持します。
-R	--recursive	ディレクトリごと再帰的にコピーします。
	--hpfp	HpFPプロトコルを使用します。
	--user=<ユーザ名>	ユーザ名を指定します。
	--mcd=<多重接続数>	多重接続数を指定します。
-h	--help	ヘルプを表示します。

hcpコマンドの実行例

利用者番号はb59999とします。
以下の操作はPCでコマンドプロンプト(ターミナル)を開いて行ってください。

- file1.txtをリモートマシンにコピーする

```
$ hcp file1.txt hpcfs.kudpc.kyoto-u.ac.jp:~/
Login as:   ←ユーザ名を入力
Passphrase for the key:   ←パスフレーズを入力
```

- ユーザ名を指定した上で、file1.txtをリモートマシンにコピーする

```
$ hcp --user=b59999 file1.txt hpcfs.kudpc.kyoto-u.ac.jp:~/
Passphrase for the key:   ←パスフレーズを入力
or
$ hcp file1.txt b59999@hpcfs.kudpc.kyoto-u.ac.jp:~/
Passphrase for the key:   ←パスフレーズを入力
```

- リモートマシンのfile1.txtをローカルマシンにfile2.txtとしてコピーする

```
$ hcp hpcfs.kudpc.kyoto-u.ac.jp:file1.dat file2.dat
Login as:   ←ユーザ名を入力
Passphrase for the key:   ←パスフレーズを入力
```

- ディレクトリ(dir1)をリモートマシンにコピーする

```
$ hcp -R dir1 hpcfs.kudpc.kyoto-u.ac.jp:~/
Login as:    ←ユーザ名を入力
Passphrase for the key:    ←パスフレーズを入力
```

- HpFPプロトコルを使用してfile1.txtをリモートマシンにコピーする

```
$ hcp --hpfp file1.txt hpcfs.kudpc.kyoto-u.ac.jp:~/
Login as:    ←ユーザ名を入力
Passphrase for the key:    ←パスフレーズを入力
```

- 多重接続数を指定してfile1.txtをリモートマシンにコピーする

```
$ hcp --mcd=1 file1.txt hpcfs.kudpc.kyoto-u.ac.jp:~/
Login as:    ←ユーザ名を入力
Passphrase for the key:    ←パスフレーズを入力
```

※mcd(多重接続数)はデフォルトで8に設定しています。

詳細なオプション、他のコマンド(hcp-ls, hmv等)のオプションの説明は--helpオプションで確認するか、オンラインドキュメントをご覧ください。

Archaea dialogによるファイル転送

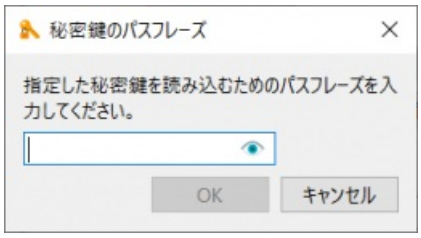
接続方法

1. Arcaea dialogを起動します。
2. 「新規サイト」をダブルクリックします。

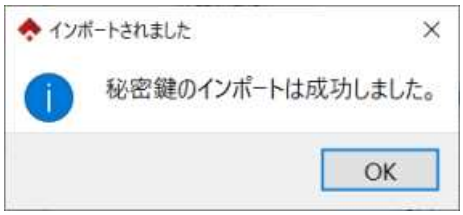
3. 以下の情報を入力し、「秘密鍵管理」をクリックします。

FQDNもしくはIPアドレス：hpcfs.kudpc.kyoto-u.ac.jp
ユーザ名：利用者番号(ID)
認証方式：公開鍵認証
プロトコルとセキュリティ：TCP もしくは HpFP

4. パスフレーズを入力し、「OK」をクリックします。



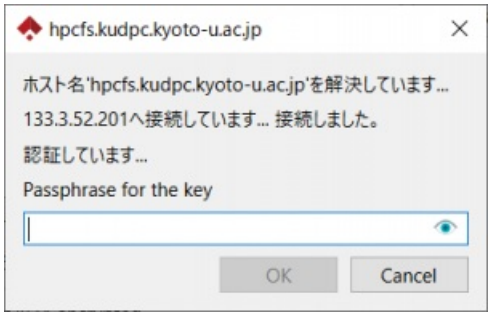
5. 「秘密鍵のインポートは成功しました。」と表示されたら「OK」をクリックします。

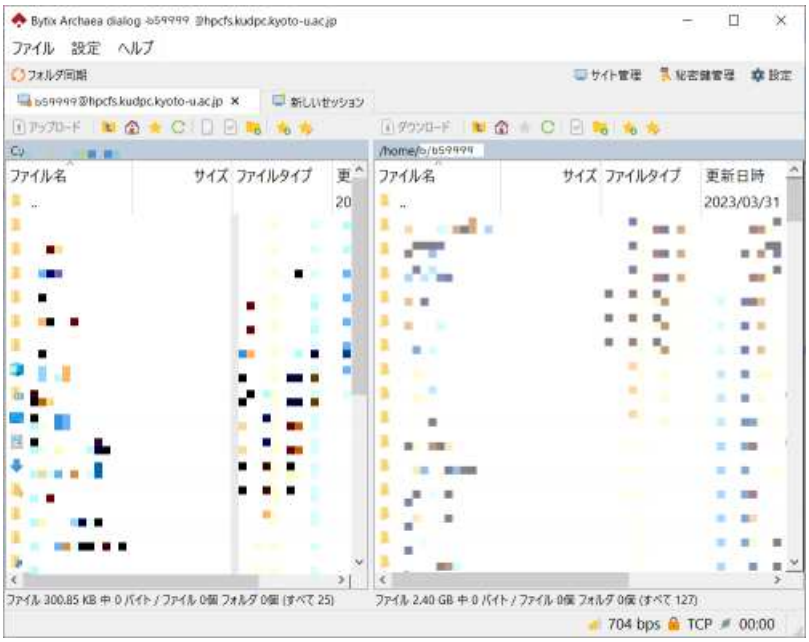


6. 登録済みの鍵一覧に先ほど登録した鍵が表示されていることを確認し、画面を閉じます。

7. サイト一覧から、先ほど設定したサイトを選択し、「接続」をクリックします。(2回目以降はこの手順から行います。)

8. 登録した鍵のパスフレーズを入力し、「OK」をクリックします。成功するとファイル一覧の画面が表示され、ファイル転送が可能になります。





補足事項

HpFPプロトコル

HpFPはUDPベースのプロトコルです。遅延・パケットロスが発生している環境の場合、HpFP(--hpfpオプション)を使用することで、高速化が期待できます。 HpFP(--hpfpオプション)を使用しない場合、TCPで接続します。セッション多重化により、scpやsftpでの転送より多少高速です。

WebSocketプロトコル

WebSocketプロトコルでの通信は許可していません。

使用するポート

HCP toolsでの通信には、874/tcp,65520/udp(--hpfpオプション使用時)を使用します。 ご利用の環境によってはポートが開いていない可能性があります。その場合は、通信できません。

ユーザあたりの接続数

ファイル転送サーバを公平にご利用いただくために、ユーザあたりのサーバへの接続数を「16」に制限しています。 --mcdオプションを用いて接続数を指定した場合の同時接続数の例は以下の通りです。

- --mcdオプションを指定しない場合
  接続数は8使用したものとみなされ、最大同時接続数は2となります。
- --mcdオプションで8を指定した場合
  接続数は8使用したものとみなされ、最大同時接続数は2となります。
- --mcdオプションで1を指定した場合
  接続数は1使用したものとみなされ、最大同時接続数は16となります。

エラー例

- 鍵、パスフレーズが異なる場合

```
Authentication failed since performing it with a challenge and the user inputs is not successful.
Authentication was failed.
```

  鍵、パスフレーズが正しいか確認してください。
- 多重接続の一部で失敗した場合

```
CH[12, primary=0] setup failed.
Error code (9, Bad file descriptor)
```

  ネゴシエーションに成功した接続で通信を行います。 通信終了までそのままお待ちください。
- 権限のないディレクトリを参照した場合

```
A signal message was received from the remote host. The session will abort in a reason specified b
Could not offer a request @ _hcp::proto::HcppSessionInitiator_:L177.
```

  ディレクトリのパスが正しいか確認してください。

コマンドの実行記録

HCP Toolsのコマンドを実行した記録は、-hcp-outオプションで出力するファイルを指定しない場合、実行ディレクトリの次のファイルに記録されます。

```
・Windowsをクライアントとして利用する場合：
    .hcp.out
・Linuxをクライアントとして利用する場合：
    _hcp.out
```

実行記録の例1

```
SRC test.txt
DST hpcfs.kudpc.kyoto-u.ac.jp:65520:/home/b/b59999/
OK 0000 FT 00000001 /home/b/b59999/test.txt
EXIT 0 REASON 0000
```

コマンドは正常（EXIT 0）に終了しています。

実行記録の例2

```
SRC test.txt
DST hpcfs.kudpc.kyoto-u.ac.jp:65520:/home/b/b59999/
OK 0000 FT 00000001 /home/b/b59999/test.txt
EXIT 72 REASON 9008
```

コマンドは認証失敗（EXIT 72）で終了しています。

終了コード（コマンド終了ステータス）の詳細は、オンラインドキュメントを参照してください。

Archaea dialogのディレクトリ移動について

~~現在のバージョン(Archaea dialog 1.0)では、ディレクトリパスを直接指定して移動する機能はありません。将来的には機能が実装される予定です。~~
ディレクトリパスを指定して移動する機能が実装されました。

マニュアル

- Archaea tools オンラインドキュメント

リンク

外部リンク

- Bytix サポートサイト
- Archaea ダウンロード（最新版）
- Archaea リリースノート

MobaXtermの利用方法

MobaXtermとは

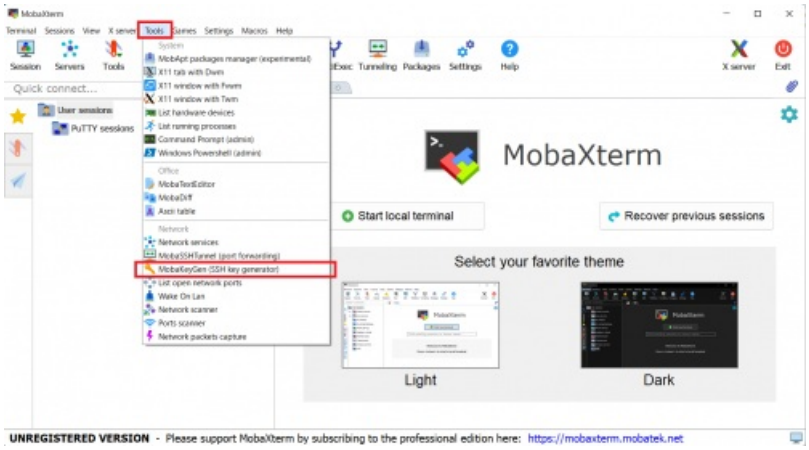
MobaXtermはSSHクライアント、X11サーバ、ネットワークツール等を備えたWindows用拡張ターミナルです。

前提条件

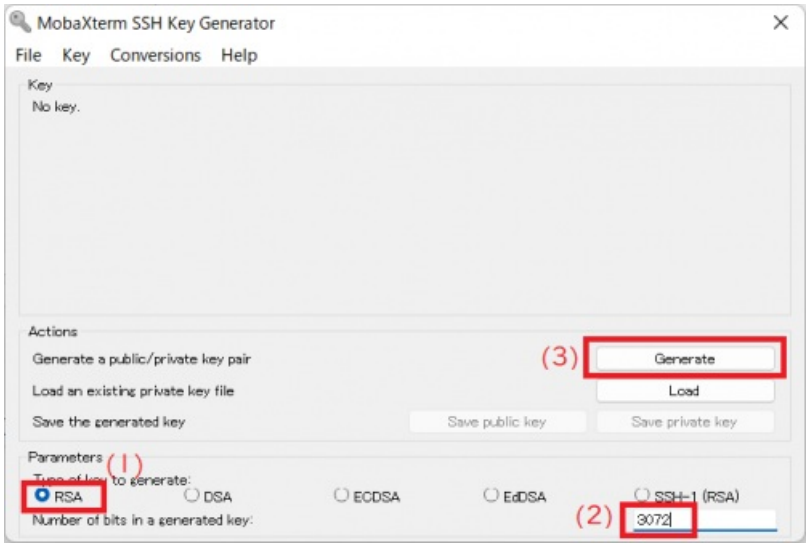
1. MobaXtermのインストールの手順でMobaXtermのインストールが完了していること。

鍵ペアの作成方法

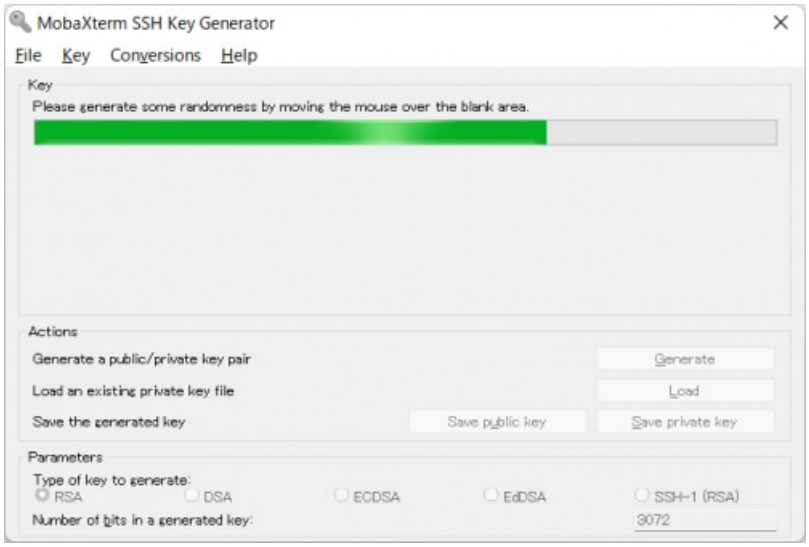
1. 「Tools」アイコンをクリックの後、「MobaKeyGen (SSH key generator)」をクリックします。



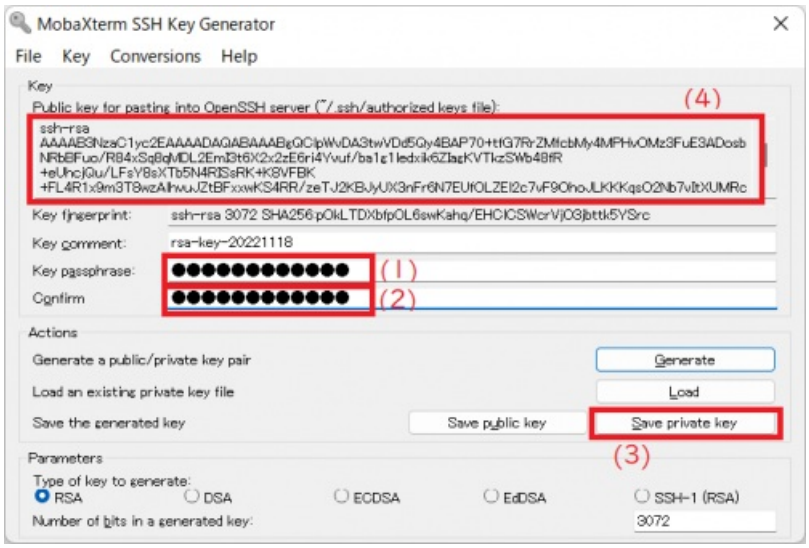
2. Type of key to generateをRSA、Number of bits a generated keyを3072に指定します。
3. 「Generate」をクリックします。



4. 空白のエリアでマウスを動かすよう表示されますので、指示にしたがってください。表示されているインジケータが右端まで進むまでの十数秒間マウスを動かす必要があります。

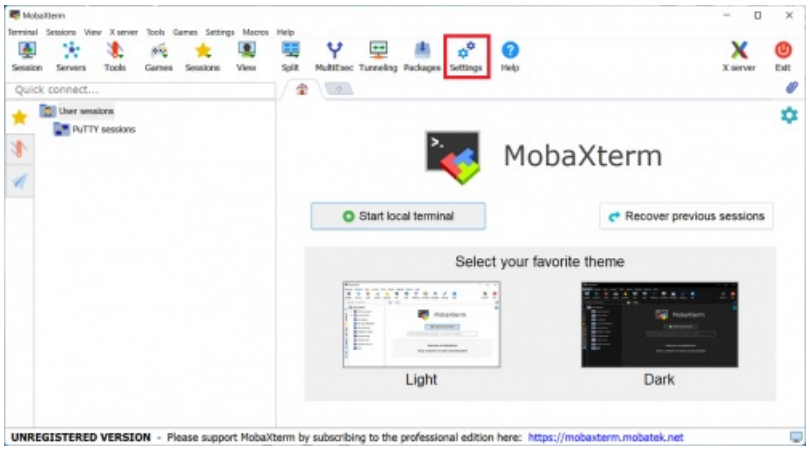


5. 鍵ペアの作成ができると画面が切り替わりますので、 パスフレーズを2か所 (Key passphrase, Confirm)入力してください。
6. 「Save private key」ボタンを押し、秘密鍵を保存します。秘密鍵が他人に渡ることのないよう、厳重に管理してください。
7. 公開鍵をコピーし、利用者ポータル から登録してください。公開鍵は1行ですが、文字数が長いため折り返して表示されています。スクロールで隠れている分も含めて全てコピーして登録する必要があります。



MobaXtermへの秘密鍵登録方法

1. MobaXtermを起動します。
2. 「Setting」をクリックの後、「SSH」をクリックします。

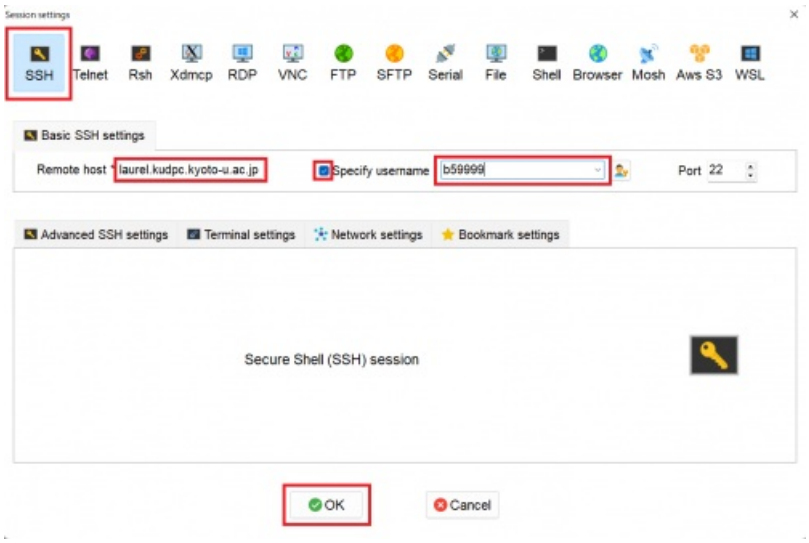


3. SSH Agentsで「Use internal SSH agent “MobAgent”」にチェックを入れます。
4. Load following keys at MobAgent startup 右側の「＋」をクリックし、鍵ペアの作成と公開鍵の登録で作成した秘密鍵を選択します。
5. 「OK」をクリックし、MobaXtermを再起動します。

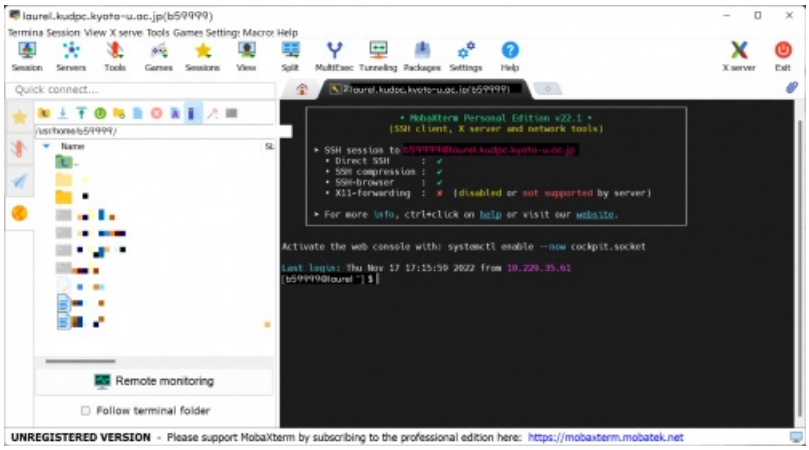
6. 起動時に秘密鍵のパスフレーズを入力し、「OK」をクリックします。
   ※以降、MobaXterm起動時に秘密鍵のパスフレーズを入力します。

接続方法

1. 「Sessions」アイコンをクリックの後、「SSH」をクリックします。
2. Remote host の欄にログインするシステムに応じたホスト名を入力します。
3. 「Specify username」にチェックを入れ、ボックスにスーパーコンピュータシステムの利用者番号(ID)を入力します。
4. 「OK」をクリックします。



5. [ユーザ名@ホスト名 ~] の形式のプロンプトが表示されればログイン成功です。

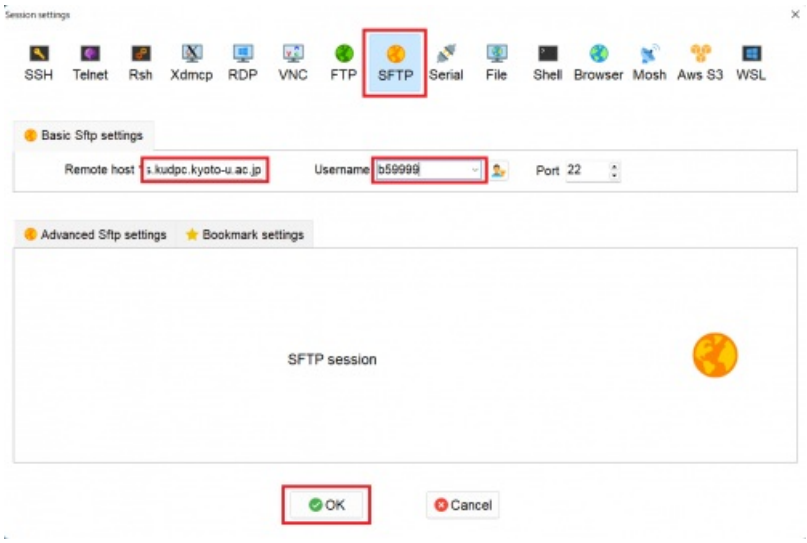


X (GUI) を利用する場合

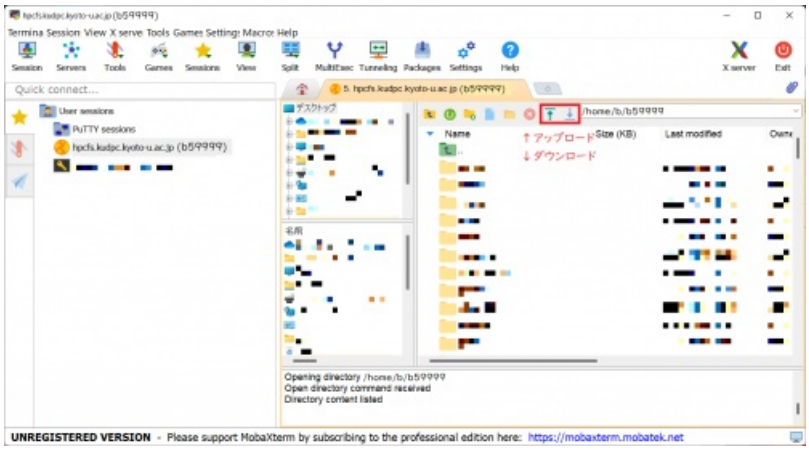
接続方法 の手順でシステムに接続してください。特別な設定は必要ありません。

ファイルの転送方法

1. 「Sessions」アイコンをクリックの後、「SFTP」をクリックします。
2. Remote host の欄にログインするシステムに応じたホスト名を入力します。
3. 「Username」ににスーパーコンピュータシステムの利用者番号(ID)を入力します。
4. 「OK」をクリックします。



5. 成功すると、スーパーコンピュータ上の自分のホームディレクトリが表示されます。上矢印でスパコン上にアップロード、下矢印でローカルマシンにダウンロードできます。



接続先

システムごとのホスト名は以下の通りです。

システム名	ホスト名	備考
システムA	camphor.kudpc.kyoto-u.ac.jp	2台のログインノードで構成。
システムB/クラウド	laurel.kudpc.kyoto-u.ac.jp	3台のログインノードで構成。
システムC	cinnamon.kudpc.kyoto-u.ac.jp	3台のログインノードで構成。
システムG	gardenia.kudpc.kyoto-u.ac.jp	2台のログインノードで構成。
アプリケーションサーバ	app.kudpc.kyoto-u.ac.jp	
ファイル転送サーバ	hpcfs.kudpc.kyoto-u.ac.jp	2台のサーバで構成。 時間制限のないSFTP、RSYNC専用のサーバ。

前システムからの移行

2022年度に更新されたシステムへ、それ以前のシステムから移行する方向けの情報です。

ホスト名、ログイン方法

ホスト名（ラウンドロビン）およびログイン方法は、前システムから変更はありません。 なお、ログインノードへラウンドロビンを経由せず直接ログインする場合は、個別のホスト名が変更となっておりますのでシステムへの接続方法を参照してください。

ログイン時にエラーで入れない場合

下記のようなメッセージが出て接続できない場合は、known_hosts情報を削除する必要があります。

```
@@@  WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!   @@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that a host key has just been changed.
The fingerprint for the RSA key sent by the remote host is
c0:30:d6:93:b2:d8:06:4a:6f:9c:d5:00:cc:c5:69:58.
Please contact your system administrator.
Add correct host key in /home/xxx/.ssh/known_hosts to get rid of this
message.
Offending RSA key in /home/xxx/.ssh/known_hosts:3
RSA host key for laurel.kudpc.kyoto-u.ac.jp has changed and you have
requested strict checking.
Host key verification failed.
```

known_host情報の削除は下記のいずれかの手順で行います。

ターミナル

- ssh-keygenコマンドを使う
  (例) laurel の known_hosts を削除する

  ```
  $ ssh-keygen -R laurel.kudpc.kyoto-u.ac.jp
  ```

- known_hostファイルを直接編集する
  1. エディタなどで `%homepath%\.ssh\known_hosts` (Windows)、 `/Users/(ユーザ名)/.ssh/known_hosts` (Mac、Linux)ファイルを開きます
  2. 中身を消去して保存します

MobaXterm

1. MobaXtermを終了します。

2. エディタなどで `%appdata%\MobaXterm\MobaXterm.ini` を開きます
3. [SSH_Hostkeys] の該当のホストの情報を削除します。

```
ssh-ed25519@22:laurel.kudpc.kyoto-u.ac.jp=0xd152edcd(以下略)
```

4. MobaXtermを起動します。

$HOME/.bashrcの初期化方法

前システムからモジュールの構成やアプリケーションの配置場所、環境変数などが変更となっています。前システムで.bashrcをカスタマイズしてご利用いただいていた方は、必要に応じて.bashrcの修正をお願いします。
また、以下のように /etc/skel/.bashrc を ホームディレクトリにコピーすることで、.bashrcを初期化することができます。
なお、ログインができない状態の場合は、お問合せフォームよりその旨お知らせいただけましたら、管理者権限でシェルの設定ファイルを初期化いたします。

- ホームディレクトリに/etc/skel/.bashrcをコピーする場合

```
$ cp /etc/skel/.bashrc $HOME
```

$HOME/.sshディレクトリについて

現システムより、SSHの公開鍵を利用者ポータルで一括して管理する運用に変更しました。これに伴い、ホームディレクトリ（$HOME）の.sshディレクトリは$HOME/DOTFILES_20221108/ 以下に移動しました。不要であれば削除してください。

データ移行

前システムに保存されていたユーザーデータは、全て自動で移行されています。

大容量ストレージ(LARGE)について

/LARGE2は/LARGE0に、/LARGE3は/LARGE1に集約されました。/LARGE2は/LARGE0へ、/LARGE3は/LARGE1へリンクを貼り、従来のパスでアクセスできるように設定しますが、将来的に設定は削除しますので、パスの更新を行ってください。

また、大容量ストレージのquota管理がGroup QuotaからProject Quotaに変更になりました。 これに伴い、容量管理はファイルが所属するグループではなく、大容量ストレージの グループのパス単位で行われます。

新システムのファイルシステム構成の詳細については、 ファイルシステムの利用 をご覧ください。

ログインノードのプロセスリミット

PCへのファイル転送中の中断などが起きにくいよう配慮し、各システムのログインノードのCPU時間とメモリ量を拡張しています。

システム	CPU時間(標準)	CPU時間(最大)	メモリ量(標準)
前システム	4時間	24時間	8GB
新システム	4時間	24時間	16GB

変更点

OS

OSがCLE/RHEL 7からRHEL 8に変更になります。

コンパイラ・ライブラリ

コンパイラは、Intel、NVIDIA HPC SDK、GNUを提供します。Crayコンパイラは提供されなくなります。

バッチジョブスケジューラ

ジョブスケジューラがPBSからSlurmに変更になります。

ジョブスクリプトのオプション比較

動作	PBS	Slurm
ジョブを投入するキューを指定する	-q *QUEUENAME*	-p *QUEUENAME*
ジョブの実行グループを指定する	-ug *GROUPNAME*	不要になりました
経過時間上限値を指定する	-W *HOUR* : *MIN*	-t *HOUR*:*MIN*
・ プロセス数を指定する ・ プロセスあたりスレッド数を指定する ・ プロセスあたりCPUコア数を指定する ・ プロセスあたりメモリサイズを指定する	-A p=*X*:t=*X*:c=*X*:m=*X*	--rsc p=*X*:t=*X*:c=*X*:m=*X*
標準出力ファイル名を指定する	-o *FILENAME*	変更なし
標準エラー出力ファイル名を指定する	-e *FILENAME*	変更なし

動作	PBS	Slurm
標準出力・エラー出力をまとめる	-j oe(標準出力に出力) / eo(標準エラーに出力)	変更なし
メール送信	-m a(ジョブ中断時) / b(開始時) / e(終了時)	--mail-type=BEGIN(開始時)/END(終了時)/FAIL(ジョブ中断時)/REQUEUE(再実行時)/ALL(全て)
メールアドレスを指定する	-M *MAILADDR*	--mail-user=*MAILADDR*
障害発生時のジョブ再実行の禁止を指定	-r n	--no-requeue

ジョブ関連コマンド比較

動作	PBS	Slurm
ジョブを投入できるキューを確認する	qstat -q	spartition
ジョブをキューに投入する	qsub	sbatch
ジョブの状態を確認する	qstat	squeue
投入したジョブをキャンセルする	qdel	scancel
ジョブの詳細情報を確認する	qs	sacct -l

環境変数比較

内容	PBS	Slurm
ジョブID	QSUB_JOBID	SLURM_JOBID
ジョブを投入したキューの名称	QSUB_QUEUE	SLURM_JOB_PARTITION
ジョブを投入したカレントディレクトリ	QSUB_WORKDIR	SLURM_SUBMIT_DIR
ジョブ実行時の割当プロセス数	QSUB_PROCS	SLURM_DPC_NPROCS
ジョブ実行時のプロセスあたりの割当スレッド数	QSUB_THREADS	SLURM_DPC_THREADS
ジョブ実行時のプロセスあたりの割当CPUコア数	QSUB_CPUS	SLURM_DPC_CPUS
ジョブ実行時のプロセスあたりの割当メモリ量上限値	QSUB_MEMORY	ー
ジョブ実行時のノード当たり配置プロセス数	QSUB_PPN	ー

ジョブスクリプト変換

pbs2slurm コマンドを利用することで、PBSの環境で使用していたジョブスクリプトのコマンドやオプションをSlurm向けに変換することができます。

書式

```
pbs2slurm input_script [output_script]
```

実行例

```
[b59999@camphor1 script]$ cat pbs.sh
#!/bin/bash
#======Option========
#QSUB -q gr19999b
#QSUB -A p=1:t=1:c=1:m=1G
#QSUB -W 12:00
#QSUB -r n
#QSUB -M kyodai.taro.1a@kyoto-u.ac.jp
#QSUB -m be
#====Shell Script====
mpiexec.hydra ./a.out

[b59999@camphor1 script]$ pbs2slurm pbs.sh slurm.sh

[b59999@camphor1 script]$ cat slurm.sh
#!/bin/bash
#======Option========
#SBATCH -p gr19999b
#SBATCH --rsc p=1:t=1:c=1:m=1G
#SBATCH -t 12:00
#SBATCH --no-requeue
#SBATCH --mail-user=kyodai.taro.1a@kyoto-u.ac.jp
#SBATCH --mail-type=BEGIN,END
#====Shell Script====
srun ./a.out
```

変換に対応するオプション

pbs2slurmコマンドでは、以下のオプションの変換に対応しています。 以下に含まれていないオプションについては、個別に修正をお願いします。

変換前	変換後	内容
#QSUB -q	#SBATCH -p	キューの指定
#QSUB -A	#SBATCH --rsc	リソースの指定
#QSUB -W	#SBATCH -t	経過時間の指定
#QSUB -N	#SBATCH -J	ジョブ名の指定
#QSUB -o	#SBATCH -o	標準出力の書き出し先の指定
#QSUB -e	#SBATCH -e	標準エラー出力の書き出し先の指定
#QSUB -m	#SBATCH --mail-type	メールの送信タイミングの指定
#QSUB -M	#SBATCH --mail-user	メールの送信先の指定

変換前	変換後	内容
#QSUB -r n	#SBATCH --no-requeue	ジョブの再実行禁止
#QSUB -J	#SBATCH -a	アレイジョブの指定
mpiexec	srun	MPIの実行(オプションがある場合は手動で削除が必要です)
mpiexec.hydra	srun	MPIの実行(オプションがある場合は手動で削除が必要です)

ログインノードの環境設定

ログインシェル

標準のログインシェルはbashに設定しています。 基本的なPATHの設定等は自動的に読み込まれるようになっていますが、個人用の起動ファイルを作成するには、.bashrc （bashの場合）等を用意してください。

ホームの.bashrc に最初から記述されている以下の内容は、bash環境の設定に必要ですので、消さないでください。

```
# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi
```

ログインシェルの変更

ログインシェルの変更は、利用者ポータルから行います。

1. 利用者ポータルにログインします。
2. 左メニューのログインシェル変更を選択してください。
3. プルダウンメニューから変更したいログインシェルを選択し、送信を押します。

シェルの起動ファイル作成時の注意点

特定のシステムだけで行いたい処理がある場合は、以下の例のようにhostnameコマンドの結果に応じた条件文を記述することで実現可能です。

.bashrc – bashの場合

```
case `hostname` in
    camphor*)
        #システムAの場合の処理
    ;;
    laurel*)
        #システムB、C、クラウドの場合の処理
    ;;
    gardenia*)
        #システムGの場合の処理
    ;;
esac
```

.tcshrc – tcshの場合

```
switch(`hostname`)
    case camphor*:
        #システムAの場合の処理
        breaksw
    case laurel*:
        #システムB、C、クラウドの場合の処理
        breaksw
    case gardenia*:
        #システムGの場合の処理
        breaksw
endsw
```

環境設定切り替えソフトウェア(Modules)の利用

modulesは、各システムでコンパイラ・ライブラリ・アプリケーションを利用する際に必要な環境設定を一括で行うことができるソフトウェアです。詳しい利用方法は Modules をご覧ください。

2023年度デフォルト環境 (2023-04-01時点)

ログインノードごとに、ログイン時に自動でロードされるモジュール環境が異なります。 それぞれ次の表のバッチ処理の環境と、コンパイラの環境がロードされます。

年度が替わる際やシステム構成が変わる際に、ロードする環境やバージョンの見直しを行います。

ログインノード	システム環境	バッチ処理環境	コンパイル環境
camphor.kudpc.kyoto-u.acjp	SysA	slurm	intel, intelmpi, PrgEnvIntel
laurel.kudpc.kyoto-u.acjp	SysB	slurm	intel, intelmpi, PrgEnvIntel
cinnamon.kudpc.kyoto-u.ac.jp	SysC	slurm	intel, intelmpi, PrgEnvIntel
gardenia.kudpc.kyoto-u.ac.jp	SysG	slurm	nvhpc, openmpi, PrgEnvNvidia

ログイン後の環境から、 クラウドシステムの環境に切り替えたい場合は次のコマンドで切り替えることができます。

```
$ module switch SysCL
```

システムからの通知メール

Slurmのエラー通知など、システムからの通知メールはシステムA、B、C、Gのローカルメールアドレス宛てに送信されます。このメールは **mutt** コマンドでも確認できますが、いち早く閲覧できるようホームディレクトリに .forward ファイルを作成し、普段利用しているメールアドレスに転送されることをお勧めしています。

.forwardファイルの作成例

```
## foo@example.com にメールを転送するよう設定
$ echo "foo@example.com" > ~/.forward
```

メッセージの日本語化

コマンドの出力や、コンパイラ翻訳時のメッセージを日本語化するには、環境変数LANGに **ja_JP.UTF-8** を設定してください。また、ご利用のSSHクライアント(PuTTYなど)の文字コードも **UTF-8** に設定してください。

```
## 環境変数LANGを設定(bashの場合)
$ export LANG=ja_JP.UTF-8

## 環境変数LANGを設定(tcshの場合)
$ setenv LANG  ja_JP.UTF-8

## manコマンドの表示が日本語になる
$ man man

man(1)                                                          man(1)

名前
       man - オンラインマニュアルページを整形し表示する。
       manpath - ユーザー個々のマニュアルページの検索パスを決める。

書式
       man  [-adfhktwW]  [-m  system]  [-p string] [-C config_file] [-M path] [-P
       pager] [-S section_list]  name ...
```

プロセスリミット

ログインノードは多数のユーザで共用するため、CPU、メモリの利用に制限を設けています。 CPU時間については、後述するコマンドで拡張が可能です。必要に応じて実施してください。

項目	初期値	最大値
CPU時間(CPU Time) / プロセス	4時間	24時間
CPUコア数 / ユーザ	4コア	同左
メモリサイズ / ユーザ	16GB	同左

bashの場合

bashを利用している場合、ulimitコマンドを使用します。

現在の設定を確認

```
$ ulimit -a
...
cpu time               (seconds, -t) 14400
...
```

最大値まで拡張

```
$ ulimit -t 86400    # CPU TIMEを最大値まで拡張
$ ulimit -a          # 確認
...
cpu time               (seconds, -t) 86400
...
```

tcshの場合

tcshを利用している場合、limitコマンド、unlimitコマンドを使用して設定の確認と最大値へ拡張ができます。

現在の設定を確認

```
$ limit
cputime      4:00:00
...
```

最大値まで拡張

```
$ unlimit  # 拡張を実施
$ limit    # 確認
cputime      24:00:00
...
```

cronの設定

定期的な処理を自動で実行させたい場合などは、cronを使用します。

複数台で構成されているログインノードのうち、システムAではcamphor31、システムBではlaurel31、システムCではcinnamon31、システムGではgardenia11のみ設定が可能です。

cronの設定（システムAの場合）

```
## cron設定が可能なcamphor31にログイン
$ ssh b59999@camphor31.kudpc.kyoto-u.ac.jp

## cronの設定内容を確認する
$ crontab -l

## cronの設定(エディタが開く)
$ crontab -e
```

環境設定ツール (modules)

概要

Modulesソフトウェアパッケージは、コンパイラ・ライブラリ・アプリケーションの利用に必要な環境設定をmoduleコマンドを実行することで、動的に切り替えて設定することができます。 異なるバージョンのコンパイラ・ライブラリ・アプリケーションを切り替えて利用する際に、簡単に環境設定を変更することができます。

利用方法

moduleコマンドの基本操作

コマンド	説明
module list	ロード済みmoduleファイルの一覧表示
module avail	使用できる全moduleファイルの一覧表示
module show [module_file_name]	moduleファイルの設定内容の表示
module load [module_file_name]	moduleファイルのロード
module unload [module_file_name]	moduleファイルのアンロード
module switch [module_file_name1] [module_file_name2]	moduleファイルの切り替え（module_file_name1→module_file_name2）

モジュールファイルの依存関係を無視した強制変更

モジュールファイル間の依存関係により module load / unload / switch のエラーが発生する場合があります。 依存関係に異常が生じる可能性をご理解の上であれば、-f オプションを使うことで、強制的にモジュールファイルのロードやスイッチをすることができます。

```
 ## ログイン直後の状態
$ module list
Currently Loaded Modulefiles:
 1) slurm/2022   2) SysB/2022   3) intel/2022.3(default)   4) intelmpi/2022.3(default)   5) PrgEnvIntel/20

## intelコンパイラを 2018.4 に切り替えようとしてもエラーになる
$ module switch intel/2018.4
Unloading intel/2022.3
  ERROR: intel/2022.3 cannot be unloaded due to a prereq.
    HINT: Might try "module unload PrgEnvIntel/2022" first.

Switching from intel/2022.3 to intel/2018.4
  ERROR: Unload of switched-off intel/2022.3 failed

## -f オプションを使用した強制的な切り替え
$ module switch intel/2018.4 -f
Unloading intel/2022.3
  WARNING: Dependent 'PrgEnvIntel/2022' is loaded

## 変更後の状況
$ module list
Currently Loaded Modulefiles:
 1) slurm/2022   2) SysB/2022   3) intelmpi/2022.3(default)   4) PrgEnvIntel/2022(default)   5) intel/2018
```

コンパイラ・ライブラリの利用方法

ロードされているmoduleファイルの確認

各プログラミング環境利用のための包括的なModulesファイルセットとして、以下の3種を用意しています。 これらのモジュールファイルにより、コンパイラの環境とMPIの環境の初期設定が行われます。

- Intelプログラミング環境 (システムA/B/C/D/Eの標準環境) : PrgEnvIntel
- NVIDIA HPC SDKプログラミング環境 (システムGの標準環境) : PrgEnvNvidia
- GNUプログラミング環境 : PrgEnvGCC

現在のモジュール環境を確認するには `module list` コマンドを使用してください。

```
 $ module list
Currently Loaded Modulefiles:
  1) XXX
```

プログラミング環境の切り替え

ログイン時はシステムのメインコンパイラ環境になっています。 以下にプログラミング環境を切り替える例を記載しています。

例　**Intel**環境から**NVIDIA**環境に切り替える場合

```
 $ module switch PrgEnvIntel PrgEnvNvidia
```

コンパイラバージョンの切り替え方法

複数バージョンが利用可能な場合、利用するコンパイラバージョンを切り替えることができます。 以下にintelコンパイラの場合の操作方法を例示しています。

例 用意されているプログラミング環境を切り替える場合

```
$ module switch PrgEnvIntel/2022 PrgEnvIntel/2018
```

例 **Intel**コンパイラの別バージョンを追加でロードする場合**(**後からロードしたほうが優先されます**)**

```
$ module load intel/Y.Y
```

例 **Intel**コンパイラのバージョンを個別に切り替える場合**(-f**が必要**)**

```
$ module switch intel/X.X intel/Y.Y -f
```

現在ロードしているバージョンの記述(上記例のintel/X.X)は省略することもできます。

```
$ module switch intel/Y.Y -f
```

アプリケーションの利用方法

MATLABを利用する場合の例

moduleコマンドを実行し、環境設定を行います。（利用したいバージョンのmoduleファイルをロードします）

```
$ module load matlab/R2020b
```

MATLABの起動コマンドを実行することができるようになります。

```
$ matlab
```

MALABのバージョンを切り替える場合（**R2020b→R2021a**）

moduleコマンドを実行し、環境設定の変更を行います。

```
$ module unload matlab/R2020b
$ module load matlab/R2021a
または
$ module switch matlab/R2021a
```

他のアプリケーションでの使い方は、利用可能なソフトウェアのページをご確認ください。

moduleファイルの作成と追加

moduleファイルをユーザ自身で作成し、利用することができます。 作成方法については、 man page をご覧ください。 作成したmoduleファイルを利用するためには、module use コマンドにmoduleファイルを格納したディレクトリを指定し、パスを追加してください。 一度追加したパスを削除したい場合は、module unuseコマンドを実行してください。 moduleファイルのパスは環境変数MODULEPATHで確認できます。

例 **module**ファイルパスの確認

```
## 作成したmoduleファイルを利用できるようにする(x12345ユーザの場合)
$ module use /home/x/x12345/mymodule

## moduleファイルのパスを確認する
$ echo $MODULEPATH
/home/x/x12345/mymodule:/opt/app/modulefiles/common:
/opt/system/app/env/intel/isv:
/opt/system/app/env/intel/oss:

## 追加したパスを削除する
$ module unuse /home/x/x12345/mymodule

## 作成したmoduleファイルを、元のパスより優先度を低くして利用できるようにする
$ module use -a /home/x/x12345/mymodule

## moduleファイルのパスを確認する
$ echo $MODULEPATH
/opt/app/modulefiles/common:/opt/system/app/env/intel/isv:
/opt/system/app/env/intel/oss:
/home/x/x12345/mymodule:
```

例 **module**ファイルの追加と利用

```
## 作成したmoduleファイルを確認する
$ ls -F /home/x/x12345/mymodule
ruby/
$ ls -F /home/x/x12345/mymodule/ruby
1.8.7  1.9.3  2.1.2

## 作成したmoduleファイルを利用できるようにする
$ module use /home/x/x12345/mymodule

## 作成したmoduleファイルが表示されることを確認する
$ module avail
--------------------------- /home/x/x12345/mymodule/ ---------------------------
ruby/1.8.7 ruby/1.9.3 ruby/2.1.2
------------------------ /opt/system/app/intel/ -------------------------
PrgEnvIntel/2022          intel/2022.3
（略）

## 作成したmoduleファイルをロードする
$ module load ruby/2.1.2e
```

リンク

Welcome to the Environment Modules Project
Man page of Modulefile

ストレージの利用

ホームディレクトリと大容量ストレージと高速ストレージ

すべてのユーザーは、各々のホームディレクトリを利用できます。また、パーソナル、グループ、専用クラスタコースを利用の方は、大容量ストレージを利用できます。 パーソナル、グループ、専用クラスタコースで計算に大規模なファイルを扱う場合は、大容量ストレージをご利用ください。 データの読み書きが多いプログラムをご利用の場合は、高速ストレージをご利用いただくことで、性能が向上する可能性があります。

ホームディレクトリ

ホームディレクトリのパスは、次の構成になっています。 自身で利用するソフトウェアのインストール先としてもご利用いただけます。

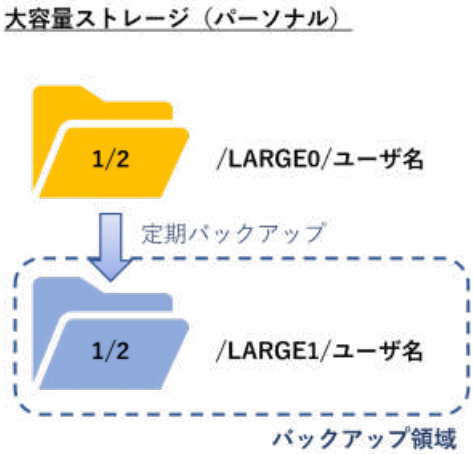
- 利用可能な容量は100GBです。(User Quota)
- 「b」の部分は利用者番号の先頭のアルファベットが入ります。
- 週に1回程度の頻度で定期的にバックアップを取得しています。

大容量ストレージ(LARGE)

パーソナル、グループ、専用クラスタコースでは、大容量ストレージを利用できます。 多数のストレージドライブで構成することで高い性能を有していますので、大容量のファイルや並列計算におけるファイルアクセスは大容量ストレージをご利用ください。

パーソナルコースの場合

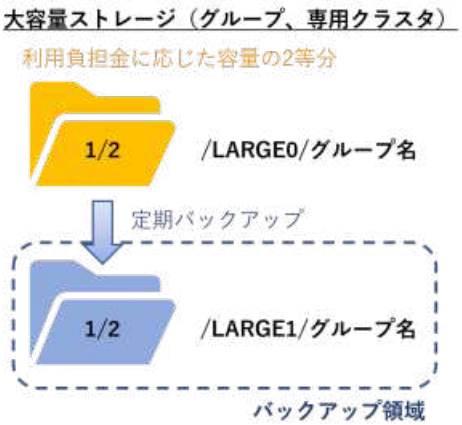
パーソナルコースで利用可能な大容量ストレージは、次の構成になっています。



- 利用可能な容量は全部で3TBになります。
- 2つのファイルシステムに分割して構成しています。
- 初期設定では、/LARGE1 をバックアップ領域としています。

グループコース、専用クラスタの場合

グループコース、専用クラスタで利用可能な大容量ストレージは次の構成になっています。グループ用のディレクトリの配下にユーザごとのディレクトリを作成するなどしてお使いください。



- 利用可能な容量はお申し込みの計算資源やストレージの追加容量に応じたサイズになります。
- 2つのファイルシステムに分割して構成しています。
- 初期設定では、/LARGE1 をバックアップ領域としています。

バックアップ設定

パーソナルコースの利用者本人またはグループコースのグループ管理者であれば、大容量ストレージのバックアップ設定を変更することが可能です。 データの冗長性を重視する場合は、バックアップを有効にし、使える容量を重視する場合はバックアップを無効に設定してください。 バックアップの頻度は、小規模な利用であれば週1回程度、数千万ファイルを持つような大規模なグループの場合は月1回程度を目標としています。

設定手順は バックアップ設定 をご覧ください。

前システムからの変更点

前システムでは、/LARGE0, /LARGE1, /LARGE2, /LARGE3 の4つで構成していましたが、今回のシステムでは /LARGE0, /LARGE1 の2つに変更になりました。 /LARGE2のデータは/LARGE0 に、/LARGE3のデータは/LARGE1 に集約しておりますが、従来のパスでファイルアクセスが可能なようにシンボリックも設定しています。

将来的にシンボリックリンクを削除する可能性がありますので、適宜パスの更新を行ってください。

```
 $ ls -ld /LARGE?
drwxr-xr-x 201 root root 12288 Nov 18 11:46 /LARGE0
drwxr-xr-x 202 root root 12288 Nov  7 18:17 /LARGE1
lrwxrwxrwx   1 root root     7 Nov  8 10:30 /LARGE2 -> /LARGE0
lrwxrwxrwx   1 root root     7 Nov  8 10:31 /LARGE3 -> /LARGE1
```

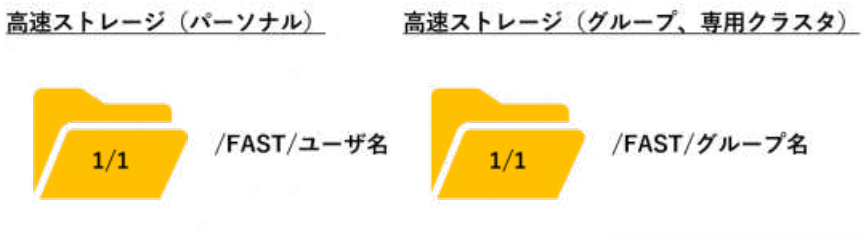
高速ストレージ(FAST)

パーソナル、グループ、専用クラスタコースでは、高速ストレージを利用できます。 高速ストレージは、全てSSDで構成されているため、高いスループットを有している一方で、容量はそこまで大きくありません。

高いスループットを活用して頂くためにも、データの長期的な保存用に使うのは避けていただき、演算の入出力にご活用ください。 計算結果の保存が必要な際には、大容量ストレージ（LARGE） に移動させる等、高速ストレージの容量と性能を活かせるよう工夫してご利用をお願いします。

2024年度はお試し期間として、パーソナルコースの方は1人あたり500GB、グループコースの方は1ノードあたり400GB〜1000GB（タイプにより変動）、ご利用いただけます。利用可能な容量の詳細は quota -p でご確認ください。

高速ストレージは次の構成になっています。



- 利用可能な容量はお申込みの計算資源や大容量ストレージの追加容量に応じたサイズになります。

バックアップについて

高速ストレージ（FAST）では、バックアップは取得しません。ご自身で適宜バックアップを行ってください。また、システム更新時にデータ移行の対象にはなりません。

使用状況の確認(quotaコマンド)

現在のディスク使用状況は **quota** コマンドで確認できます。

- kbytes: 使用中のファイル容量（KB）
- quota: ファイル容量／数の制限値（ソフトリミット）

- limit: ファイル容量／数の絶対的制限値（ハードリミット）
- grace: 制限値越え（ソフトリミット超過）の許容期間
- files: 使用中のファイル数

HOMEディレクトリの使用状況

```
$ quota -u b59999
Disk quotas for user b59999 (uid 59999):
  Filesystem       kbytes          quota          limit grace    files     quota     limit grace
       /home     17859980     100000000     110000000     -    69218   1000000   1100000     -
```

大容量ストレージ(LARGE)/高速ストレージ(FAST)の使用状況確認

LARGEはProject Quotaで容量を制限しています。 quota コマンドに -p オプションを指定して実行してください。

```
$  quota -p
Disk quotas for project gr19999 (pid 19999):
  Filesystem       kbytes        quota        limit grace    files    quota    limit grace
     /LARGE0   2088651052   8000000000   8050000000     -    90543 16000000 16100000     -
     /LARGE1   1967559464   8000000000   8050000000     -    90544 16000000 16100000     -
     /FAST   762012   500000000   505000000     -    4544 16000000 16100000     -
```

プログラムの実行

ジョブスケジューラについて

スーパーコンピュータシステム(以下、スパコン)は、ユーザのプログラムを「ジョブ」という単位で実行します。 本センターのスパコンでは、Slurm というジョブスケジューラを用いて、ジョブの制御を行っています。 一部機能をカスタマイズしているため、標準的なSlurmとは使い勝手に若干の差異がありますので、ご注意ください。

ジョブスケジューラは、「キュー」と呼ばれる仮想的な入れ物でジョブを管理しており、システムにバッチ処理を依頼する際にはキューを指定する必要があります（キューにジョブを投入する）。 スパコンでは、お申し込み頂いたサービスコースごとにキューを用意しておりますので、そのキューを使用してジョブを実行いただくことになります。 Slurm の公式マニュアルでは、キューのことをパーティションと表現されていますが、本マニュアルではキューで統一致します。

また、申し込みいただいたサービスコースによって利用できる計算資源の量が決められています。詳細は 計算資源の割当 のページをご覧ください。

ジョブの実行形式について

ジョブを実行する形式として、「会話型処理」と「バッチ処理」があります。具体的なジョブの投入方法は個別ページをご覧ください。

- 会話型処理
  - インタラクティブ(会話的)な入出力操作や、GUI操作を必要とするプログラムを利用する場合に使用します。
  - ジョブの実行にあたっては、ジョブ投入時のシェルが常時起動している必要があります。
- バッチ処理
  - インタラクティブな入出力操作を必要としないプログラムを実行する場合に使用します。
  - 特に、実行時間の長いものや、並列数が大きいプログラムはバッチ処理を利用してください。
  - 実行したい一連の処理を記述した「ジョブスクリプト」を作成する必要があります。

バッチ処理

ジョブ実行の流れ

1. ジョブスクリプトの作成
2. ジョブの投入
3. (必要であれば)ジョブの状態の確認
4. (必要であれば)ジョブのキャンセル

ジョブスクリプト

ジョブスクリプトは、原則的にシェルスクリプトと同じ形式です。 ジョブスクリプトはSlurmのジョブ投入オプションを記述したオプション領域と、実行するプログラムを記述したユーザプログラム領域から構成されます。 ジョブ実行時に自動的に設定される環境変数についてはこちらをご参照ください。

4プロセスx8スレッドのハイブリッド並列の例

ジョブスクリプトの一例を以下に示しますので、まずはスクリプトの内容を見て全体像を把握してください。 次節以降に、それぞれのより詳細な情報を記載しています。

```
#!/bin/bash
#============ Slurm Options ===========
#SBATCH -p gr19999b             # ジョブキュー(パーティション)の指定。投入したいキュー名に適宜変更が必要
#SBATCH -t 1:00:00              # 経過時間の指定(1時間を指定する例)。
#SBATCH --rsc p=4:t=8:c=8:m=8G  # 要求リソースの指定(4プロセスx8スレッドのハイブリッド並列の例)。
#SBATCH -o %x.%j.out            # ジョブの標準出力ファイルの指定。%xはジョブ名、%j はジョブIDに置換され
#============ Shell Script ============

# (任意) set -x を指定するとジョブスクリプトの実行経過を把握しやすくなります。
set -x

# MPIのプロセス数やOMP_NUM_THREADS等の環境変数は--rscオプションの値を元に自動で設定されます。
# 必要に応じてsrun コマンドの引数や環境変数により確保したリソースの範囲で上書き指定が可能です。
srun ./a.out

## ジョブスクリプトに関する補足 ##
# 「#」で始まる行あるいは、行中の「#」以降は、コメントとして扱われます。#SBATCH で始まる行のみ例外的にs
# ジョブ実行時のカレントディレクトリは、ジョブ投入時のカレントディレクトリに自動で移動します。
# ジョブ投入時に設定されている環境変数は、ジョブ実行時に継承されます。
```

サンプルスクリプト

ジョブスクリプトのサンプルを公開していますので、参考にしてください。

実行種別	サンプルファイル

実行種別	サンプルファイル
逐次実行	ダウンロード
スレッド並列	ダウンロード
プロセス並列(Intel MPI)	ダウンロード
ハイブリッド並列	ダウンロード

Slurmオプション

ジョブスクリプトの Slurm Options の部分で「#SBATCH」に続けて指定します。

主要なオプション

オプション	意味	指定例
-p *QUEUE*	キューの指定 (必須項目)	-p gr19999b
-t *HOUR:MINUTES:SECONDS*	実行時間の上限設定	-t 24:0:0
--rsc p=*PROCS*:t=*THREADS*:c=*CORES*:m=*MEMORY* or --rsc g=*GPU*	リソースの指定。詳細はこちら	--rsc p=4:t=8:c=8:m=8G or --rsc g=1
-o *FILENAME*	標準出力の保存先の指定。指定可能な特殊文字は公式マニュアルを参照。	-o result.out
-e *FILENAME*	標準エラー出力の保存先の指定。指定可能な特殊文字は公式マニュアルを参照。	-e result.err
-J *JOBNAME*	ジョブ名の指定。 指定しない場合は、ジョブスクリプト名がジョブ名になります。	-J ReplaceJobName
--comment=*Comment*	コメントの指定	--comment=ThisIsComment
-a *ARRAY_SPEC*	アレイジョブの指定。詳しくはこちら	-a 1-5
-d *TYPE:JOBID*	ジョブの実行順序の指定。詳しくはこちら	-d afterok:999999
--no-requeue	障害発生時のジョブ再実行の禁止を指定	--no-requeue
--mail-user=*MAILADDR*	メールアドレスの指定	--mail-user=bar@sample.com

オプション	意味	指定例
--mail-type=*TYPE*	イベント通知の指定 BEGIN, END, FAIL, REQUEUE, ALLを必要に応じて指定	--mail-type=BEGIN,END

その他のオプションや、オプションの詳細は 公式マニュアル をご参照ください。 また、ジョブ投入する際に使用できないオプションについても、必要に応じてご確認ください。

プログラムの実行方法(重要)

プログラムを計算ノード上で動作させるためには、逐次プログラム、MPIプログラムに関わらず、ジョブスクリプトのプログラムを 実行する箇所に 必ず ***srun*** コマンドを使用する必要があります。

srunコマンドのオプションのうち、代表的なものを以下に示します。 その他のオプションや、オプションの詳細は、公式マニュアルをご参照下さい。

基本オプション

オプション	機能
-n *PROCS*	起動するプロセス数を指定する。指定がない場合は、--rscオプションの pの値が使用されます。
-c *CORES*	プロセスあたり確保するCPUコア数を指定する。指定がない場合は、--rscオプションのcの値が使用されます。
--ntasks-per-node=*PROCS_PER_NODE*	ノードあたりのプロセス数を指定する。他のパラメータと合わせて、ノード当たりのプロセス数を制御したい場合に使用して下さい。

使用可能なキューの確認

投入可能なジョブのキューを確認するためには、spartitionコマンドを使用します。

spartitionコマンド

投入可能なジョブのキュー名、Rmin/Rstd/Rmax、経過時間の標準値/上限値を表示します。Rmin/Rstd/Rmaxについては、こちら をご参照ください。

```
$ spartition
Partition  State   Rmin  Rstd  Rmax    DefTime     MaxTime
gr19999g   UP         0    64    64   01:00:00  1-00:00:00
```

ジョブの投入

キューにジョブを投入するためには、sbatchコマンドを使用します。

sbatchコマンド

```
$ sbatch sample.sh
Submitted batch job 20
```

- 作成したジョブスクリプトファイルを、コマンドの最後に続けて入力します
- ジョブスクリプトの実行がシステムに依頼され、ジョブIDが表示されます
- ジョブ投入時にオプションを指定した場合、ジョブスクリプト内のオプションは上書きされます

ジョブ投入時メッセージ

```
$ sbatch sample.sh
sbatch: cli_filter/accms_resource_req: convert_rsc_option: Updated the number of cores from c=10 to c=4
Submitted batch job 21
```

上記のメッセージは、要求されたメモリサイズに基づき、コア数を変更したというメッセージです。エラーではないので、ジョブは投入されます。

ジョブの状態の確認

投入したジョブの情報を表示するためには、squeueコマンド、sacct、qsコマンドコマンドを使用します。

squeueコマンド

現在キューに登録されているジョブの情報を表示します。

```
$ squeue
JOBID PARTITION     NAME     USER ST   TIME  NODES NODELIST(REASON)
    1  gr19999b interact   b59999  R   0:33      1 no0001
```

- ジョブ実行中のみ情報が表示されます。
- オプションについては、公式マニュアルをご参照下さい。

ジョブ実行に関する備考(REASON)

ジョブが実行待ち状態の場合は、NODELIST(REASON)にその理由が表示されています。 代表的なものを以下に示します。 その他の理由については、公式マニュアル をご参照下さい。

REASON	意味
Resources	現在空きリソースがありません
QOSJobLimit	同時実行可能数の上限に達しています

sacctコマンド

アカウンティングデータベースにあるジョブの情報を表示します。過去のジョブの情報も表示することができます。 システムへの負荷が高いコマンドであるため、機械的な繰り返しの実行は控えてください。

```
$ sacct -X
JobID           JobName  Partition    Account  AllocCPUS      State ExitCode
------------ ---------- ---------- ---------- ---------- ---------- --------
1               test.sh   gr19999b     b59999          8  COMPLETED      0:0
2               test.sh   gr19999b     b59999          8  COMPLETED      0:0
```

- 標準ではコマンド発行日に実行されたジョブの一覧が表示されます。

主要なオプション

オプション	意味	指定例
-j	指定したジョブの統計情報を表示する。	sacct -j 1234556
-X	ジョブステップを考慮せず、ジョブの割り当て自体に関連する統計のみを表示する	sacct -X
-l	ジョブに関するすべての情報を表示する。	sacct -l / sacct -Xl
-S	指定した時刻以降のジョブを表示する。	sacct -S 2022-10-01

その他のオプションは、公式マニュアルをご参照下さい。

qsコマンド

ジョブの情報を表示します。

```
$ qs
 QUEUE     USER     JOBID          STATUS  PROC  CORE    MEM    ELAPSE(    limit)
 gr19999b  b59999   1              RUN        4     1  4570M  00:00:07( 01:00:00)
```

- ジョブ実行中のみ情報が表示されます。

ヘッダ	概要
QUEUE	キュー名
USER	ユーザ名
JOBID	ジョブID
STATUS	ジョブの状態
PROC	プロセス数

ヘッダ	概要

CORE	プロセスあたりのコア数
MEM	プロセスあたりのメモリ量
ELAPSE	ジョブの経過時間
limit	ジョブの経過時間の制限値

キューの状態の確認

グループのキューの状況を確認するためには、qgroupコマンドを使用します。

qgroupコマンド

グループのキュー全体の情報、利用者ごとの統計情報を表示します。

```
$ qgroup
 QUEUE    SYS |   RUN  PEND OTHER | ALLOC ( MIN/ STD/ MAX)
-------------------------------------------------------------
 gr19999b  B  |     0     0     0 |     0 (   0/224/224)

 QUEUE    USER     |   RUN(ALLOC)  PEND(REQUEST) OTHER(REQUEST)
-------------------------------------------------------------
 gr19999b b59999   |     1(    4)     0(     0)     0(     0)
```

-l オプションを指定すると、グループのキュー全体の情報、利用者ごとの統計情報に加えて、ジョブごとの情報が表示されます。

```
 $ qgroup -l
 QUEUE    SYS |   RUN  PEND OTHER | ALLOC ( MIN/ STD/ MAX)
-------------------------------------------------------------
 gr19999b  B  |     0     0     0 |     0 (   0/224/224)

 QUEUE    USER     |   RUN(ALLOC)  PEND(REQUEST) OTHER(REQUEST)
-------------------------------------------------------------
 gr19999b b59999   |     1(    4)     0(     0)     0(     0)

 QUEUE    USER     JOBID    | STAT  SUBMIT_AT        | RSC:core | PROC CORE    MEM       ELAPSE
-------------------------------------------------------------------------------------------------
 gr19999b b59999   1        | RUN   2024-06-06 16:51 |        4 |    4    1  4570M     01:00:00
```

キューごとの状況

ヘッダ	概要

ヘッダ	概要
QUEUE	キュー名
SYS	システム名
RUN,PEND,OTHER	ジョブの本数
ALLOC	割り当てコア数
MIN	キューの最低保証コア数
STD	キューの標準コア数
MAX	キューの最大コア数

ユーザごとの状況

ヘッダ	概要
RUN(ALLOC)	実行中のジョブ数と割り当て資源量(コア数換算)
PEND(REQUEST)	実行待ちのジョブ数と要求資源量(コア数換算)
OTHER(REQUEST)	上記以外の状態のジョブ数と要求資源量(コア数換算)

ジョブごとの状況

ヘッダ	概要
STAT	ジョブの状態
SUBMIT_AT	ジョブの投入日時
RSC:core	資源量(コア数換算)
PROC	プロセス数
CORE	プロセスあたりのコア数
MEM	プロセスあたりのメモリ量
ELAPSE	ジョブの経過時間の制限値

ジョブのキャンセル

投入したジョブをキャンセルするためには、scancelコマンドを使用します。

scancel コマンド

```
$ scancel 20
```

- ジョブIDを引数に指定します。
- オプションは、公式マニュアルをご参照下さい。

投入したジョブを全てキャンセルする方法

投入したすべてのジョブをキャンセルする場合は、以下の方法でキャンセルすることができます

```
## ユーザ名を指定して削除する場合
$ scancel -u b59999

## キュー名を指定して削除する場合
$ scancel -p gr19999b

## 実行中のステータスを指定して削除する場合
$ scancel -t pending
```

会話型処理

会話型実行のためのコマンド

会話型実行はSlurmのインタラクティブバッチ機能を使用して行います。簡易に実行するためのツールとして ***tssrun*** コマンドを用意しています。 tssrun コマンドの後ろに実行したいプログラムおよび引数を指定することでジョブを実行することが可能です。 ジョブ実行時に自動的に設定される環境変数についてはこちらをご参照ください。

コマンド	説明	実行例	備考
tssrun	計算ノードでプログラムを実行する	tssrun ./a.out	

コマンドオプション

オプション	説明	指定例
-p *QUEUENAME*	バッチ処理用キューの指定	tssrun -p gr19999b ./a.out
-t *HOURS*:*MINUTES*:*SECONDS*	経過時間上限値の指定（単位は時：分：秒）	tssrun -p gr19999b -t 1:0:0 ./a.out
--rsc p=*PROCS*:t=*THREADS*:c=*CORES*:m=*MEMORY* or --rsc g=*GPU*	ジョブ割当リソース量の指定。詳細はこちら	tssrun -p gr19999b --rsc p=4:t=8:c=8:m=2G ./a.out or tssrun -p gr19999b --rsc g=1 ./a.out
--x11	計算ノードでGUIプログラムを実行する	tssrun -p gr19999b --x11 xeyes

- 指定例に、*gr19999b* と記載の部分はご自身のキュー名に変更していただく必要があります。
- コマンドを入力すると、いくつかのメッセージが表示された後、プログラムの実行が開始されて結果が表示されます。
- 会話型実行用の計算ノードが混雑している場合、メッセージ表示後すぐに実行が開始されないことがあります。
- パーソナルコース、グループコース用のキューで会話型実行を行うことも可能です。

ジョブの投入

実行例**1:** システムBのMPIプログラムの場合

```
$ tssrun -p gr19999b --rsc p=4 ./a.out
salloc: Granted job allocation 102362
salloc: Waiting for resource configuration
salloc: Nodes cnode3 are ready for job
My name is cnode3. My rank is 1.
My name is cnode3. My rank is 2.
My name is cnode3. My rank is 3.
My name is cnode3. My rank is 0.
salloc: Relinquishing job allocation 102362

exit code: 0
```

実行例**2:** GUIプログラムを実行する場合

```
$ tssrun -p gr19999b --x11 xeyes
salloc: Granted job allocation 102366
salloc: Waiting for resource configuration
salloc: Nodes <計算ノード名> are ready for job
（Xアプリケーションが起動する）
```

ジョブ情報の確認

tssrunコマンドによる会話型実行では、ジョブ情報を確認することが可能です。

コマンド	オプション	説明	実行例
sacct	-j *JOBID*	会話型実行のジョブ情報を表示する	sacct -j 102362

コマンドとオプションに続けて、ジョブIDを指定する必要があります。ジョブIDは、tssrunコマンド実行時に表示される `salloc: Granted job allocation XXXXXX` のXXXXXXの部分です。

会話型実行の終了後、24時間以上経過したものについては表示できません。

会話型実行が異常終了した場合のメッセージ

会話型実行が異常終了した場合、ジョブ情報中に終了理由が表示されます。

- メモリ制限による終了 (ノード全体のメモリの圧迫によるもの)

```
Can't malloc: Cannot allocate memory
```

- 経過時間制限による終了

```
slurmstepd: error: *** STEP 102368.0 ON <計算ノード名> CANCELLED AT 2022-11-04T15:34:36 DUE TO TIM
```

計算資源の割り当て

キューに割り当てられる計算資源

スーパーコンピュータシステムでは、お申し込み頂いたサービスコースごとにキューを用意しており、そのキューを指定してジョブを実行いただけます。

すべてのキューには、最小資源量、標準資源量、最大資源量の3つの制限値が設定されており、キューで利用できる計算資源の量が決められています。

制限項目	意味
最小資源量(Rmin)	キューに対して常に確保されるCPUコア数
標準資源量(Rstd)	キューに投入されたジョブが利用できるCPUコア数の最大値
最大資源量(Rmax)	キュー全体で利用できるCPUコア数の最大値

最小資源量

最小資源量は、キューに対して常に確保されるCPUコア数のことで、最低保証資源量ともよばれます。
システム定常稼働時には、システムの混雑状況によらず、最小資源量として設定された数のCPUコアを利用できることが保証されます。

標準資源量

標準資源量は、キューに投入されたジョブが利用できるCPUコア数の最大値です。
ジョブ投入時に --rsc オプションを使うことでジョブに割り当てるリソース量を指定できますが、標準資源量を超えるリソース量を要求した場合にはエラーが発生します。

```
$ sbatch --rsc p=400 ./jobscript.sh
Too many processors requested. Job not submitted.
```

最大資源量

最大資源量は、キュー全体で利用できるCPUコア数の最大値です。
システムの資源に空きがある場合、キューに投入されたジョブは順次実行され、複数のジョブが同時に実行状態となることもあります。しかし、キュー内のすべてのジョブの使用コア数の総和が最大資源量を超えることはできません。キュー全体で利用しているCPUコア数が最大資源量の値に達すると、たとえシステムの資源に空きがあったとしても、ジョブは実行待ち状態となります。

計算資源の初期値と--rscオプションで指定可能な最大値

【訂正事項】
- システムAの「プロセスあたりのメモリ量」の記載に誤記がありましたので修正しました．(2024/05/29)
  - 初期値: (誤)1142M -> (正) 1071M

- 最大値: (誤)120G -> (正) 117G

オプション	説明	システム A 初期値	最大値	システム B 初期値	最大値	システム C 初期値	最大値
p	プロセス数	1	標準資源量 (c=1の場合)	1	標準資源量 (c=1の場合)	1	標準資源量 (c=1の場合)
t	プロセスあたりのスレッド数	1	112(224)	1	112(224)	1	112(224)
c	プロセスあたりのコア数	1	112	1	112	1	112
m	プロセスあたりのメモリ量 (単位：M, G)	1071M	117G	4571M	500G	18392M	2011G

オプション	説明	システムG 初期値	最大値	クラウド 初期値	最大値
p	プロセス数	1	標準資源量 (c=1の場合)	1	1
t	プロセスあたりのスレッド数	1	64(128)	1	36(72)
c	プロセスあたりのコア数	1	64	1	36
m	プロセスあたりのメモリ量 (単位：M, G)	8000M	500G	14222M	500G
g	GPU数	1	標準資源量	-	-

補足情報

ハイパースレッディングについて

tの（）内は、ハイパースレッディングを有効にする場合の最大値です。なお、ハイパースレッディングを有効にする場合は、t=c×2 となるように指定してください。

コア数、メモリ量の自動補完及び自動補正について

- mの値を指定しない場合は、プロセスあたりのメモリ量の初期値×プロセスあたりのコア数が自動的に設定されます。
- mの値を指定した場合、それに比例してコア数を要求したと見なされます。具体的には以下のとおりです。
  - システムBを使用するジョブで、p=1:t=4:c=4:m=9142M (初期値4571M x2)と指定した場合、cオプションが4コア、mオプションが2コアを要求します。この時より多くのコアを確保するオプションのコア数がcの値として採用されるため、1プロセスあたり4コアのジョブとなります。なお、1プロセスあたり4コアを使用するジョブとなりますが、メモリは指定した資源量(9142M)が最大値となります。
  - システムBを使用するジョブで、p=1:t=2:c=2:m=13713M (初期値4571M x3)と指定した場合、cオプションが2

コア、mオプションが3コアを要求します。この場合は、cオプションで要求したコア数よりも、mオプションで要求されるコア数のほうが大きいため、1プロセスあたり3コアのジョブとなります。

要求するGPU数の指定について

システムGを利用する場合は、ptcmの各オプション または gオプションの何れかで資源を要求することができます。

- ptcmの各オプションで資源を要求する場合は、16コアあたり1GPUが自動的に確保されます。
- gオプションを用いてGPUを要求する場合は、以下のような挙動になります。
  - g=1とした場合は、1GPUの確保と同時に、p=1:c=16:m=128000M と同様のパラメータが自動的に設定されます。
  - g=2とした場合は、2GPUの確保と同時に、p=2:c=16:m=128000M と同様のパラメータが自動的に設定されます。
    - 1プロセスで2GPUを使用する場合は、ジョブスクリプトに記載するsrunコマンドに、-n 1 オプションを指定することで実現できます。(例: srun -n 1 ./a.out)
    - 逆にプロセス数を増やしたい場合は、srun の -c オプションでプロセスあたりのコア数も併せて調整してください。(例: srun -n 9 -c 1 ./a.out)

グループコースキューの計算資源

グループコースキューは、グループコースを申し込み頂いた方のグループに個別に用意されるキューです。

最小資源量、標準資源量、最大資源量は以下のように決められます。

制限項目	設定値
最小資源量	コースタイプと標準資源量から算出された値
標準資源量	申請書に記載された値
最大資源量	標準資源量の1倍～2倍の範囲で変動

大規模ジョブコースキューの計算資源

大規模ジョブコースキューは、大規模ジョブコースを申し込み頂いた方に個別に用意されるキューです。

最小資源量、標準資源量、最大資源量は以下のように決められます。

制限項目	設定値
最小資源量	標準資源量と同じ値
標準資源量	申請書に記載された値
最大資源量	標準資源量と同じ値

リソースを確保するオプション (--rsc) の具体例

ここでは、--rsc オプションの働きを、例示しながら説明します。 なお、バッチジョブに限らず、tssrun で指定する --rsc

オプションでも同様です。

前提

ここではシステムBを想定しますが、他のシステムでも--rscオプションの法則は同様です。

- システムBの1ノード当たりのスペック
  - CPU：112コア
  - メモリ：1コアあたり4096M (ノード全体では 500G)
  - ハイパースレッディング：1コアあたり2スレッドまで

--rscオプションの各パラメータ

- p : MPIプロセス数。デフォルト1
- t : プロセスごとのOpenMPスレッド数。デフォルト1
- c : プロセスごとのコア数。デフォルト1
- m : プロセスごとのメモリ量。デフォルト4096M

例1：OpenMP 4スレッドの計算を実行

```
#SBATCH --rsc p=1:t=4:c=4:m=8G
```

- CPUの物理コアに1スレッドずつ割り付けて実行します。
- 使用できるメモリ量は8GBとなります。

例2：MPI 60並列の計算を実行

```
#SBATCH --rsc p=60:t=1:c=1
```

- CPUの物理コアに1プロセスずつ割り付けて実行します。
- プロセスあたりのメモリ量の指定がされていないため、各プロセスの利用できるメモリは4096M(初期値)となります。

例3：MPI 6並列かつ1プロセスあたりOpenMP 4スレッド、メモリ20Gを必要とするハイブリッド並列

```
#SBATCH --rsc p=6:t=4:c=4:m=20G
```

- CPU4コアに1プロセスずつ割り付け、CPUの物理コアに1スレッドずつ割り付けて実行します。
- ただし、計算資源の確保としては以下のような動きとなりますのでご注意ください。
  - 本センターでは、CPUコアにメモリを紐付ける形で管理をしており、システムBの場合は4096MB毎にCPU1コアを 確保する必要があります。

- この例では、プロセスあたりのメモリ量を20GB(m=20G)と指定しており、20,000M/4,096M=4.88 コア から、1プロセスあたり最低5コアを割り当てる必要があります。
- ジョブスケジューラが確保するCPUコアは、--rscオプションで指定するCPUコア数と、メモリを確保するために 必要となるCPUコア数の何れか大きい値が採用されます。 この例の場合、メモリを確保するために必要なCPUコア数が、--rscオプションで指定するコア数を上回っており、 1プロセスで使用するCPUコア数は5コアとして資源確保が行われます。 なお、プロセスの割付は、--rscオプションで記載した、c=4,t=4の値を元に行われますので、残りの1コアは 遊休資源となり、実際に使われることはありません。

例4：ハイパースレッディングを利用してOpenMP 224並列を実行する

```
#SBATCH --rsc p=1:t=224:c=112:m=500G
```

- p=1,c=112により1ノードの112物理コアを確保し、t=224により1個の物理コアに2スレッドを起動します。
- m=500Gにより、1ノードで利用できる全メモリ（500GB）を確保しています。
- なお、mの値を指定しないと、利用できるメモリはデフォルトの4096MBになってしまいます。

ジョブ実行のヒント

環境変数

環境変数の設定と参照

環境変数の設定にはexportコマンドを使用し、環境変数を参照する場合は変数名の先頭に「$」を付与して記述します。

- 環境変数の設定

```
#書式　環境変数名=値; export 環境変数名
LANG=en_US.UTF-8; export LANG
```

- 環境変数の参照

```
echo $LANG
```

ジョブ実行時に設定される環境変数

環境変数名	意味
$SLURM_CPUS_ON_NODE	コア数/ノード
$SLURM_DPC_CPUS	タスクごとの物理コア数
$SLURM_CPUS_PER_TASK	タスクごとの論理コア数
$SLURM_JOB_ID	ジョブID (アレイジョブの場合は $SLURM_ARRAY_JOB_ID を使用してください)
$SLURM_ARRAY_JOB_ID	アレイジョブ実行時の親となるジョブID
$SLURM_ARRAY_TASK_ID	アレイジョブ実行時のタスクID
$SLURM_JOB_NAME	ジョブ名
$SLURM_JOB_NODELIST	ジョブに割り当てられたノード
$SLURM_JOB_NUM_NODES	ジョブに割り当てられているノード数
$SLURM_LOCALID	ノード内の実行中のノードのインデックス
$SLURM_NODEID	ジョブに割り当てられたノードに対する相対的なインデックス
$SLURM_NTASKS	ジョブのプロセス数

環境変数名	意味
$SLURM_PROCID	ジョブに対するタスクのインデックス
$SLURM_SUBMIT_DIR	サブミットディレクトリ
$SLURM_SUBMIT_HOST	送信元ホスト

ジョブ投入時に使用できないオプション

以下のオプションはsbatchコマンドおよびジョブスクリプトの #SBATCH の指示文で指定できませんのでご注意ください。

オプション				
--batch	--clusters(-M)	--constraint(-C)	--contiguous	--core-spec(-S)
--cores-per-socket	--cpus-per-gpu	--cpus-per-task(-c)	--distribution(-m)	--exclude(-x)
--exclusive	--gpu-bind	--gpus(-G)	--gpus-per-node	--gpus-per-socket
--gres	--gres-flags	--mem	--mem-bind	--mem-per-cpu
--mem-per-gpu	--mincpus	--nodefile(-F)	--nodelist(-w)	--nodes(-N)
--ntasks(-n)	--ntasks-per-core	--ntasks-per-gpu	--ntasks-per-node	--ntasks-per-socket
--overcommit(-O)	--oversubscribe(-s)	--qos(-q)	--sockets-per-node	--spread-job
--switches	--thread-spec	--threads-per-core	--use-min-nodes	
--get-user-env	--gid	--priority	--reboot	--uid

/tmp 領域について

本センターのスーパーコンピュータシステムでは、/tmpの領域を一時的なデータの書き込み先として利用できます。 細かいファイルI/Oを伴うプログラムは、/tmpを利用したほうが高速に処理できるケースがあります。

/tmpはジョブごとにプライベートな領域として用意しますので、他のジョブとファイルが 混在しないように作りこんでありますので、ぜひご活用ください。

なお、/tmp の領域は、ジョブの終了時に自動で削除されますので、/tmpに書き込んだファイルを 残すためには、ジョブスクリプトに /homeや/LARGE0,/LARGE1 にファイルをコピーする 記述を含めて頂く必要があります。あとから取り出すことはできませんので、ご注意ください。

/tmpを使用する具体例

- プログラムで指定が可能な場合は、ファイルの書き込み先を /tmp に指定する。
- 繰り返し読み込むファイルをプログラム実行開始前に /tmp に配置する。
- 相対PATHでファイルにアクセスするプログラムや入力ファイルを、/tmp に配置したうえで実行する。

利用可能な/tmpの容量

各システムで利用可能な/tmpの容量は、 `プロセス数 x プロセスあたりのコア数 x システム毎のコアあたりの容量(次表参照)` で求めることができます。

システム名	コアあたりの容量
システムA	2.4G
システムB	8.9G
システムC	8.9G
システムG	15.6G
クラウド	94G

例えば、システムBで4プロセス(1プロセスあたり 8コア) のジョブを投入した場合は `4 x 8 x 8.9` から、**284.8GB** の割当が行われます。

ジョブの実行順序を指定する

すでに投入されたジョブの実行状況に合わせて、これから投入するジョブの実行開始のタイミングを制御することができます。 具体的には、ジョブ投入オプション欄に以下の内容を追記することで、実現することができます。

```
#SBATCH -d <実行順序の指定コマンド>:<すでに投入済みのジョブID>
```

例えば、ジョブスクリプトAの実行が正常に終了した場合に、ジョブスクリプトBを続けて実行したい場合は、 以下の手順で実現することができます。

1. ジョブスクリプトAをジョブに投入します。
2. 1.で投入したジョブのジョブIDが「200」の場合、ジョブスクリプトBは以下のようになります。 *

```
#!/bin/bash
#============= SBATCH Directives =======
#SBATCH -p gr19999b
#SBATCH -t 2:0:0
#SBATCH --rsc p=4:t=8:c=8:m=8G
#SBATCH -d afterok:200
#SBATCH -o %x.%j.out
#============= Shell Script ============
srun ./a.out
```

3. ジョブスクリプトBをジョブに投入します。

なお、実行順序の指定は以下の4つから選ぶことができます。

実行順序の指定コマンド	意味
after	指定したジョブの開始後にジョブを実行する。
afterany	指定したジョブの終了後にジョブを実行する。
afterok	指定したジョブが正常終了した場合にジョブを実行する。なお、指定したジョブが異常終了した場合は実行されずに終了します。
afternotok	指定したジョブが異常終了した場合にジョブを実行する。なお、指定したジョブが正常終了した場合は実行されずに終了します。

アレイジョブの実行

アレイジョブ機能を用いることで、パラメータが異なる複数のジョブを1つのジョブスクリプトで実行することができます。 具体的には、ジョブ投入オプション欄に以下の内容を追記することで、実現することができます。

```
#SBATCH -a <start_num>-<end_num>[option]
```

例えば、./a.out に ジョブスクリプトと同一ディレクトリに配置した、1.data, 2.data, 3.data を渡して解析を行う ジョブを実行したい場合は、以下のようなジョブスクリプトを書くことによって、実現することができます。

```
#!/bin/bash
#============ SBATCH Directives =======
#SBATCH -p gr19999b
#SBATCH -t 2:0:0
#SBATCH --rsc p=4:t=8:c=8:m=8G
#SBATCH -a 1-3
#SBATCH -o %x.%A_%a.out
## %xはジョブ名、%AはアレイジョブID、%aはアレイタスクIDに置換されます。
#============ Shell Script ============
srun ./a.out ${SLURM_ARRAY_TASK_ID}.data
```

なお、[option] には以下オプションを設定することができます。

[option]に記載する内容	実行内容
:[0-9]	指定した数値毎に実行する。例えば 「1-5:2」とした場合は、1,3,5が実行される。
%[0-9]	指定した数値が同時実行数の最大値として設定される。例えば「%2」とした場合は、同時実行数の最大値は「2」となる。

1台の計算ノードで複数のプログラムを同時に実行する方法

1つのジョブの中で、複数のプログラムを同時に実行させることによって、計算資源を有効に利用することができます。複数のプログラムを1つのジョブとして同時に実行させるためには、シェルスクリプト中に複数の実行コマンドを記載し、そのシェルスクリプトを実行するようにします。

※本手法の対象は、逐次プログラムもしくはOpenMPや自動並列化機能でスレッド並列化したプログラムです。
※複数のMPIプログラムを1つのジョブとして同時に実行する場合は、複数のプログラムを同時に実行する方法(MPMD)をご参照下さい。

各スクリプトの例を以下に記します。

逐次実行(1プログラムあたり1スレッド)

sbatchコマンドで実行するジョブスクリプト(逐次)

※下記のスクリプトは、プログラムを4つ同時に実行する場合のサンプルです

```
#!/bin/bash
#============= SBATCH Directives =======
#SBATCH -p gr19999b
#SBATCH --rsc p=4:t=1:c=1:m=3G
#SBATCH -o %x.%j.out
#============= Shell Script ============
srun ./multiprocess.sh
```

- キューを -p オプションで指定します。
- 使用するリソースを --rsc オプションで下記の通り指定します。
  - 引数pには使用するプロセス数。
    - 同時に実行したいプログラム数を指定してください。ここでは4となります。
  - 引数cには利用するプロセス（プログラム）あたりに使用するコア数。
    - プログラムはシングルスレッド実行ですので、1を指定します。
  - 引数tには利用するプロセス（プログラム）あたりに使用するスレッド数。
    - プログラムはシングルスレッド実行ですので、1を指定します。
  - 引数mには利用するメモリプロセス（プログラム）辺りの容量。
- srunコマンドに続けて実行するシェルスクリプトのパスを記載します。

ジョブスクリプト中で実行するシェルスクリプト（逐次）

```
#!/bin/bash
case $SLURM_PROCID in
    0) ./a1.out ;;
    1) ./a2.out ;;
    2) ./a3.out ;;
    3) ./a4.out ;;

esac
```

- リソースを確保すると、それぞれのプロセスごとにランク番号が付与されます。
- プロセスが実行されると、環境変数 SLURM_PROCID から割り当てられたランク番号を読み出すことが可能となります。
- ランク番号を用いて処理分岐を行うシェルスクリプトを作成することで、実行するプログラムをプロセスごとに分

けることが可能となります。 これによって、複数のプログラムを1つのジョブとして同時に実行する事が可能となります。

OpenMP実行

sbatchコマンドで実行するジョブスクリプト(OpenMP)

※下記のスクリプトは、4スレッド実行のプログラムを8つ同時に実行する場合のサンプルです

```
#!/bin/bash
#============ SBATCH Directives =======
#SBATCH -p gr19999b
#SBATCH -t 2:0:0
#SBATCH --rsc p=8:t=4:c=4:m=3G
#SBATCH -o %x.%j.out
#============ Shell Script ============
srun ./multiprocess.sh
```

- キューを -p オプションで指定します。
- 使用するリソースを --rsc オプションで下記の通り指定します。
  - 引数pには使用するプロセス数。
    - 同時に実行したいプログラム数を指定してください。ここでは8となります。
  - 引数cには利用するプロセス（プログラム）あたりに使用するコア数。
    - プログラムは４スレッド実行ですので、4を指定します。
  - 引数tには利用するプロセス（プログラム）あたりに使用するスレッド数。
    - プログラムは4スレッド実行ですので、4を指定します。
  - 引数mには利用するメモリのプロセス（プログラム）辺りの容量。
- srunコマンドに続けて実行するシェルスクリプトのパスを記載します。

ジョブスクリプト中で実行するシェルスクリプト（**OpenMP**）

```
#!/bin/bash
case $SLURM_PROCID in
    0)  ./aaa.out ;;
    1)  ./bbb.out ;;
    2)  ./ccc.out ;;
    3)  ./ddd.out ;;
    4)  ./eee.out ;;
    5)  ./fff.out ;;
    6)  ./ggg.out ;;
    7)  ./hhh.out ;;

esac
```

- リソースを確保すると、それぞれのプロセスごとにランク番号が付与されます。
- プロセスが実行されると、環境変数 SLURM_PROCID から割り当てられたランク番号を読み出すことが可能となります。
- ランク番号を用いて処理分岐を行うシェルスクリプトを作成することで、実行するプログラムをプロセスごとに分

けることが可能となります。 これによって、複数のプログラムを1つのジョブとして同時に実行する事が可能となります。

ジョブスクリプトで「#SBATCH --rsc t=4」 を指定したことで、ジョブスケジューラによって1プロセスあたり4スレッド使用する設定が自動的に行われます。 そのため、上記シェルスクリプトでは各プログラムを4スレッドで実行されます。

プロセスごとにスレッド数を変化させたい場合は、 下記のように **OMP_NUM_THREADS={**使用したいスレッド数**}** をプロセスごとに定義します。

```
#!/bin/bash
case $SLURM_PROCID in
    0) export OMP_NUM_THREADS=1; ./aaa.out ;;
    1) export OMP_NUM_THREADS=2; ./bbb.out ;;
    2) export OMP_NUM_THREADS=2; ./ccc.out ;;
    3) export OMP_NUM_THREADS=3; ./ddd.out ;;
    4) export OMP_NUM_THREADS=3; ./eee.out ;;
    5) export OMP_NUM_THREADS=4; ./fff.out ;;
    6) export OMP_NUM_THREADS=4; ./ggg.out ;;
    7) export OMP_NUM_THREADS=4; ./hhh.out ;;

esac
```

複数のプログラムを同時に実行する方法(MPMD)

sbatchコマンドで実行するジョブスクリプト**(MPI)**

※下記のスクリプトは、4プロセスを必要とするMPIプログラム(1プロセスあたり4コア、4スレッド)を 3つ同時に実行する場合のサンプルです。

```
#!/bin/bash
#============ SBATCH Directives =======
#SBATCH -p gr19999b
#SBATCH -t 2:0:0
#SBATCH --rsc p=12:t=4:c=4:m=3G
#SBATCH -o %x.%j.out
#============ Shell Script ============
srun --multi-prog ./multiprocess.conf
```

- キューを -p オプションで指定します。
- 使用するリソースを --rsc オプションで下記の通り指定します。
  - 引数pには使用するプロセス数。
    - 同時に実行したいプロセス数の合計(4プロセスを使用するMPIプログラムを3つ同時に実行したい場合は、4×3から12を指定)
  - 引数cには利用するプロセス（プログラム）あたりに使用するコア数。
    - プログラムは４スレッド実行ですので、4を指定します。
  - 引数tには利用するプロセス（プログラム）あたりに使用するスレッド数。
    - プログラムは4スレッド実行ですので、4を指定します。
  - 引数mには利用するメモリのプロセス（プログラム）辺りの容量。

- srunコマンドには以下のオプションを指定します。
  - --multi-prog には以下で作成するMPIプログラムの実行に関する内容を記述した設定ファイル(multiprocess.conf)へのパスを記載。

設定ファイル**(multiprocess.conf)**

```
## 4プロセスを使用してMPIプログラムを実行
0-3  ./aaa.out
4-7  ./bbb.out
8-11 ./ccc.out
```

- 1列目には、MPIプログラムが使用するプロセスIDのレンジを記載して下さい。(プロセスIDは、0始まりであることに注意)
- 2列目には、1列目で指定したプロセスIDで実行するMPIプログラムのパス(絶対パス、相対パスの何れにも対応)を記載して下さい。

なお、使用するプロセス数はMPIプログラムごとに変更することができます。 例えば、a1.outというプログラムでは4プロセスを使用し、a2.outというプログラムでは8プロセスを使用したいと いった場合は、以下のような設定ファイルを作成することで実現することが可能です。 ただし、1プロセスあたりのコア数、スレッド数、メモリ量については、--rscオプションで指定した値が自動的に 設定されるため、MPIプログラムごとにカスタマイズをすることはできません。

```
## 4プロセスを使用してMPIプログラムを実行
0-3  ./aaa.out
## 8プロセスを使用してMPIプログラムを実行
4-11 ./bbb.out
```

クラウドシステムの利用

このページでは、クラウドシステムの利用方法について説明します。

概要

クラウドシステムは、商用のクラウドサービスを用いて小規模な計算機クラスタを構成し、 本センターに設置したスーパーコンピュータシステムのストレージやジョブスケジューラと 接続することで、本センターに設置したスーパーコンピュータシステムと同様にジョブの 実行を行うことが可能なシステムです。

スーパーコンピュータシステムの利用者番号をお持ちであればどなたでもご利用いただけます。

システム構成

クラウドシステムで使用する計算機の仕様は以下のとおりです。 なお、クラウドシステムのノード数は需要に応じて変動します。

計算ノード仕様

現在使用可なノード

2023年6月15日の保守後より、以下のベアメタルインスタンスを使用しています。 利用可能なノード数は、負荷状況に合わせて変動します。 大規模な構成ではないため、1ジョブ当たり1ノードの規模を上限としています。

項目	内容
ノード数	変動
プロセッサ	Intel Xeon Gold 6354 3.0GHz 18コア
プロセッサ数 (コア数)	2 (36コア/ノード)
アーキテクチャ	x86-64
演算性能	3.45TFlops/ノード
メモリ容量	512GByte
ネットワーク	50Gbps Ethernet

過去に使用していたノード

2023年4月1日〜6月13日までは、以下のベアメタルインスタンスを使用していました。 リソース不足が生じた際には、このインスタンスを使用する場合があります。

項目	内容
ノード数	変動

項目	内容
プロセッサ	Intel Xeon Gold 6154 3.0GHz 18コア
プロセッサ数 (コア数)	2 (36コア/ノード)
アーキテクチャ	x86-64
演算性能	3.45TFlops/ノード
メモリ容量	384GByte
ネットワーク	25Gbps Ethernet

利用方法

スーパーコンピュータのログインノードにログインした後、以下のmodule コマンドでクラウドシステムを利用する環境に切り替えることでクラウドシステムへのジョブ投入が可能です。

```
$ module switch SysCL
```

ジョブ実行

システムA/B/C と同じXeonプロセッサを搭載したノードでプログラムを実行するため、再コンパイルすることなく、同じプログラムを利用することが可能です。 ジョブキュー（パーティション）の構成や、ノード当たりのコア数、メモリ容量に違いはありますが、それ以外は同じ使い勝手でご利用いただけます。

詳細な情報は、プログラムの実行をご参照ください。

クラウドシステムのキュー構成

クラウドシステムは以下のようなキュー構成としています。 eoキューはデバッグ用として短時間のジョブに限定しています。 soキューは小規模なジョブを実行することが可能です。 大規模なキューについては、状況に応じて提供することがあります。

逐次プログラムのジョブスクリプトの例

```
 #!/bin/bash
#============= Slurm Options ===========
#SBATCH -p eo              # ジョブキュー(パーティション)の指定。投入したいキュー名に適宜変更が必要。
#SBATCH -t 1:00:00         # 経過時間の指定(1時間を指定する例)
#SBATCH --rsc p=1:t=1:c=1  # 要求リソースの指定
#SBATCH -o %x.%j.out       # ジョブの標準出力ファイルの指定
#============= Shell Script ============
set -x

srun ./a.out
```

MPIプログラムのジョブスクリプトの例 (8プロセス)

```
 #!/bin/bash
#============ Slurm Options ===========
#SBATCH -p eo              # ジョブキュー(パーティション)の指定。投入したいキュー名に適宜変更が必要。
#SBATCH -t 1:00:00         # 経過時間の指定（1時間を指定する例）
#SBATCH --rsc p=8:t=1:c=1  # 要求リソースの指定
#SBATCH -o %x.%j.out       # ジョブの標準出力ファイルの指定
#============ Shell Script ============
set -x

srun ./a.out
```

利用可能なキュー

キュー名	利用可能なユーザ
eo	全ユーザ
so	パーソナルコース，グループコース，専用クラスタコースに所属するユーザ

キュー毎の利用可能な計算資源

説明	eoキュー 初期値	最大値	soキュー 初期値	最大値
プロセス数 (--rsc p=*X*)	1	36	1	36
プロセスあたりのスレッド数 (--rsc t=*X*)	1	36	1	36
プロセスあたりのコア数 (--rsc c=*X*)	1	36	1	36
プロセスあたりのメモリ量(--rsc m=*X*) (単位：M, G)	13G	500G	13G	500G
経過時間 (-t)	1時間	1時間	1時間	7日
ユーザあたりの同時実行数	1	1	1	1

/tmp 領域について

本センターのスーパーコンピュータシステムでは、/tmpの領域を一時的なデータの書き込み先として利用できます。 細かいファイルI/Oを伴うプログラムは、/tmpを利用したほうが高速に処理できるケースがあります。

/tmpはジョブごとにプライベートな領域として用意しますので、他のジョブとファイルが 混在しないように作りこんでありますので、ぜひご活用ください。

なお、/tmp の領域は、ジョブの終了時に自動で削除されますので、/tmpに書き込んだファイルを 残すためには、ジョブスクリプトに /homeや/LARGE0,/LARGE1 にファイルをコピーする 記述を含めて頂く必要があります。あとから取り出すことはできませんので、ご注意ください。

/tmpを使用する具体例

- プログラムで指定が可能な場合は、ファイルの書き込み先を /tmp に指定する。

- 繰り返し読み込むファイルをプログラム実行開始前に /tmp に配置する。
- 相対PATHでファイルにアクセスするプログラムや入力ファイルを、/tmp に配置したうえで実行する。

利用可能な/tmpの容量

クラウドシステムで利用可能な/tmpの容量は、 `プロセス数 x プロセスあたりのコア数 x 94GB` で求めることができます。

例えば、4プロセス(1プロセスあたり 8コア) のジョブを投入した場合は `4 x 8 x 94` から、**3,008GB** の割当が行われます。

クラウドシステムの利用に関する補足

ホームディレクトリや大容量ストレージへのアクセスについて

クラウドシステムにおいても、システムB等のオンプレミスのシステムと同様に、ホームディレクトリ($HOME)や 大容量ストレージ(/LARGE)をマウントしていますので、同じPATHでファイルにアクセスすることができます。

しかし、クラウドシステムと大学にあるストレージとの間には、ネットワークの距離や帯域の制約 があるため、ストレージ性能を完全に引き出すことはできません。

I/Oが少ないプログラムについては、システムBとの使い勝手の差は少ないと思いますが、 I/Oが多いプログラムは、前節で紹介した/tmpを有効活用してください。

巨大ファイルの扱いについて

巨大なファイルを/homeや/LARGE0/1に書き込むと、応答が悪くなりやすいため、 効率的なファイル転送をお願いします。

以下に、特定のディレクトリは以下を1つのファイルにまとめるtarコマンドの例を紹介します。

```
 ## rsyncコマンドの例(圧縮あり(-z))
$ rsync -za /tmp/target-dir/ /LARGE0/gr19999/remote-dir/

## tar+gzip のコマンド例
$ cd /tmp
$ tar -zcvf target-dir archive-name.tar.gz; cp archive-name.tar.gz /LARGE0/gr19999/archive-name.tar.gz

## tar+zst のコマンド例(多くの場合gzipよりも高速かつ高圧縮)
$ cd /tmp
$ tar -Izstd -cvf target-dir archive-name.tar.zst; cp archive-name.tar.gz /LARGE0/gr19999/archive-name.tar
```

ノード間のMPI性能について

ノード間は50Gbps のEthernet のため、スーパーコンピュータの性能として考えるとあまり高くはありません。 確保できる計算リソースも多くないため、ジョブ当たりのノードを1ノードに限定しています。

可視化サーバの利用

可視化サーバについて

スーパーコンピュータの可視化環境の改善を目的として，2018年12月より試験的に提供を始めたものです．

システム構成

京都大学 学術情報メディアセンターが提供する可視化サーバは下記のとおりです．

可視化ノード1 (gp-0002)

性能諸元	
CPU	Intel(R) Xeon(R) Silver 4110 2.10GHz (8コア) x 2
メモリ	768GB
搭載GPU	NVIDIA Quadro GV100 (32 GB HBM2) x 2

利用の流れ

可視化サーバは，システムA，B，C，Gのログインノードから利用することが可能です． システムへの接続方法 を参考にログインノードへログインしてください．
可視化サーバを利用するにあたっての申請は特になく，スパコンユーザであればどなたでもご利用頂けます．

可視化サーバでアプリケーションを実行するためのコマンド

可視化サーバを用いてアプリケーションを実行する際には，ログインノードにログインし，tssrunコマンドを用いて実行する必要があります．
tssrun コマンドに -gpu オプションおよび実行したいコマンドを指定することで，可視化サーバでプログラムが実行することができます．

実行方法

コマンド	説明	備考
tssrun -gpu	可視化サーバでGUIプログラムを実行する	FastX 等のXサーバソフトウェアでのみ利用可能．

- システム全体に関する注意事項
  - コマンドを入力すると，いくつかのメッセージが表示された後，プログラムの実行が開始されます．
  - 通常の tssrun （-gpu無し）の場合と異なり，可視化サーバでのプログラム起動前に新規ログインと同様の処理が実施されます．このため，$HOME/.bash_profile, .bashrc, .tcshrc等が改めて評価される点にご注意ください． また，モジュール環境についても再設定されるため，コマンド実行時の環境変数が正確に再現されない点にご注意ください．PATHやLD_LIBRARY_PATHの順序や定義状況に差が出る場合があります．
  - 可視化サーバの受け入れ上限値を超えると，混雑している旨を表示しプログラムが終了します．時間を空けて

再度お試しください．

実行例

```
$ module load mathematica
$ tssrun -gpu mathematica
[VGL] NOTICE: Automatically setting VGL_CLIENT environment variable to
[VGL]    10.11.0.9, the IP address of your SSH client.
（Xアプリケーションが起動する）
```

コマンドオプション

可視化サーバを用いてプログラムを実行する際に，tssrunコマンドで利用可能なオプションは以下のとおりです．

オプション	要否	説明
-gpu	必須	可視化サーバを利用してプログラムを実行するための宣言
-t HOUR:MINUTES	任意	経過時間上限値の指定（単位は時：分） 経過時間の標準は1時間(1:00)となっており，最長24時間(24:00)まで指定可能．

可視化サーバを用いてプログラムを実行する際の制限

可視化サーバを用いてプログラムを実行する際には，以下のような制限が適用されます．

- ノードの共有
  可視化サーバは，現在1台のノードにて構成されています．
  契約しているコースに関係なく，すべてのユーザがノードを共有して利用する形態となっており， 可視化サーバの受け入れ上限値を超えると，混雑している旨を表示しプログラムが終了します．
- 利用できるCPUコア数
  1ユーザが同時に利用できるCPUコア数は，**4**コア となっており変更することはできません．
- 利用できるメモリ量
  1ユーザが同時に利用できるメモリ量は，**64GB** です．予約システムを用いることで，**720GB** まで上限を拡張することが出来ます．
- プログラムの経過時間制限
  可視化サーバを用いてプログラムを実行する場合，経過時間が標準の制限(1時間)を超えるとプログラムは強制終了されます．
  -t オプションを指定することで時間を上限(24時間)まで増やすことができます．また，予約システムを用いることで， **72**時間 まで上限を拡張することが可能です．

可視化サーバの利用予約について

2019年12月より，可視化サーバの「予約」サービスを開始しています． 「予約」を行うことで，以下の通り可視化サーバの利用条件を拡大することが可能です．

	予約なし	予約あり
利用できるメモリ量	64GB(固定)	最大720GB　※予約時に30GB単位で指定が可能
最大実行時間	24時間	72時間(3日間)
GPUボードの専有利用	なし	あり(24時間経過後から予約可能)

※ 予約を行うことによって拡張される利用条件は **2023年8月8日** 現在のものです．今後の利用状況によって変更することがありますので予めご承知おきください．

予約手順

1. 利用者ポータル にログインします．

2. 「サービス申請」を押下し，「可視化サーバ予約」を選択します．

3. 「利用する可視化サーバ」ならびに，予約するメモリ量，GPUの利用種別を選択し，「次へ」を押下します．

4. 利用可能な日程一覧が表示されますので，開始時刻と終了時刻を選択してください．

5. 青く塗りつぶされた予約期間が正しいことを確認して，下部の「予約確定」を押下します．

6. 下記の画面に遷移すると予約完了となります．予約開始時間になりましたら利用の流れを参考にプログラムを実行して下さい．

コンパイラ/MPIライブラリ

コンパイラ

+ ： 利用可能
- ： 利用不可

コンパイラ	システムA	システムB/C	システムG	クラウドシステム
Intel	+	+	-	+
NVIDIA HPC SDK	-	-	+	-

ライブラリ

+ ： すべてのユーザが利用可能
AU ： 学術研究機関限定で利用可能
- ： 利用不可

メッセージ通信ライブラリ (MPI)

コンパイラ	システムA	システムB/C	システムG	クラウドシステム
Intel MPI	+	+	-	+
OpenMPI	+	+	+	-

数値計算ライブラリ

コンパイラ	システムA	システムB/C	システムG	クラウドシステム
Intel oneMKL	+	+	-	+
NAG	-	AU	-	-
IMSL	-	AU	-	-

Intelコンパイラ クラシック

利用環境

利用できるバージョン・システム

バージョン	モジュールファイル名	システムA	システムB/C	システムG	クラウドシステム	備考
2024.0	intel/2024.0	+	+	-	+	2024年4月導入. Intelコンパイラ + MKL + TBB
2023.2	intel/2023.2	+	+	-	+	2024年4月導入. Intelコンパイラ + MKL + TBB
2023.2-rt	intel/2023.2-rt	+	+	-	+	2024年4月導入. ランタイムライブラリ
2023.1 (default)	intel/2023.1	+	+	-	+	2023年8月導入. Intelコンパイラ + MKL + TBB
2023.1-rt	intel/2023.1-rt	+	+	-	+	2023年8月導入. ランタイムライブラリ
2022.3	intel/2022.3	+	+	-	+	2022年11月導入. Intelコンパイラ + MKL + TBB
2022.3-rt	intel/2022.3-rt	+	+	-	+	2022年11月導入. ランタイムライブラリ

注：Intelコンパイラの個別のバージョンではなく、Intel OneAPI の包括的なバージョンで記載しています。

+ ： すべてのユーザが利用可能
- ： 利用不可

システムA、B、C、クラウドシステムでは、システムにログインした時点でIntelコンパイラがデフォルトで設定されています。システムG は、動作検証をしていません。

```
$ module list
Currently Loaded Modulefiles:
x) slurm x) SysB/2022  x) intel/2022.3 x) PrgEnvIntel/2022
```

コンパイラのバージョンは、上表に記載のデフォルトのバージョンが設定されています。 Intelコンパイラのバージョンを切り替えたい場合は、PrgEnvIntel ごと切り替える必要があります。 以下のようにmoduleコマンドを実行してください。

```
$ module switch PrgEnvIntel/2022 PrgEnvIntel/2023
```

ログイン時に自動で環境設定を行いたい場合は、ログインシェルの起動ファイルに必要なmoduleコマンドを記述してください。詳細は 環境設定 をご覧ください。 moduleコマンドの詳細は Modules をご覧ください。

コンパイル方法

コマンド

言語	コマンド	実行形式
C	icc	icc [オプション] ファイル名
C++	icpc	icpc [オプション] ファイル名
Fortran	ifort	ifort [オプション] ファイル名

オプション

オプション名	説明
-o FILENAME	オブジェクトファイルの名前を指定します。
-mcmodel=medium	2Gbyteを超えてメモリを利用できるようになります。
-shared-intel	インテルが提供するライブラリをすべて動的にリンクします。
-fpic	位置に依存しないコードを生成します。
-qopenmp	OpenMP指示子を有効にしてコンパイルします。
-qmkl	MKLライブラリをリンクします。
-parallel	自動並列化を行います。
-O0 / -O1 / -O2 / -O3	最適化のレベルを指定します（デフォルトは-O2）。
-fast	プログラムの実行速度が最大になるように最適化します。-fast オプションにより、次のオプションが自動で付与されます。 `-ipo, -O3, -no-prec-div, -static, -fp-model fast=2 -xHost`
-ip	単一ファイル内で、手続き間の処理を最適化します。
-ipo	複数ファイル間で、手続き間の処理を最適化します。コンパイルに要する時間が大幅に増加します。
-qopt-report	実施した最適化についての情報を表示します。

オプション名	説明
-xHost	コンパイルするホスト上のプロセッサで利用可能な最上位の命令セット向けのコードを生成します。
-xCORE-AVX512 -xCORE-AVX2	Intelプロセッサ向けに，指定した命令セットに対応した最適化コードを生成します。
-static-intel	インテルが提供するライブラリを静的にリンクします。静的リンクすることで、プログラム起動時の動的ライブラリ探索の負荷を下げる効果があります。
-Bstatic	-Bstatic オプションに続くコマンドライン上のすべてのライブラリを静的にリンクします。ただし、-Bdynamic オプションが途中で見つかった場合は、それ以降は動的にリンクします。
-Bdynamic	-Bdynamic オプションに続くコマンドライン上のすべてのライブラリを動的にリンクします。ただし、-Bstatic オプションが途中で見つかった場合は、それ以降は静的にリンクします。

2GB以上のメモリを使う場合のオプション

2GB以上のメモリを使うプログラムをコンパイルする場合は、以下のオプションを指定してください。

`-mcmodel=medium -shared-intel`

コンパイル例

逐次プログラム

```
$ icc test.c      # C言語の例
$ icpc test.cpp   # C++の例
$ ifort test.f90      # Fortranの例
$ tssrun ./a.out  # 実行
```

自動並列化の利用

```
$ ifort -parallel test.f90
$ tssrun --rsc p=1:t=4:c=4 ./a.out # 並列数4を指定して実行
```

OpenMPの利用

OpenMPは、プログラムの並列化のためのオープン規格です。 ソースコードに#pragma ompで始まる指示を書き込み、所定のオプションをつけてコンパイルするだけで、コンパイラに自動で並列化を行わせることができます。

OpenMPへの指示を書き込んだソースコードをコンパイルするには、-qopenmpオプションをつけます。

```
$ icc -qopenmp test.c
```

コンパイルしたプログラムを実行する際、--rscオプションでtとcに並列数を指定すると、その並列数でプログラムが実行されます。

```
$ tssrun --rsc p=1:t=8:c=8 ./a.out # 並列数8を指定して実行
```

Coarrayの利用

Coarrayは、Fortran 2008で追加された、プログラム並列化に関する機能です。 以下のようにコンパイルすることで、Coarrayを有効化することができます。 なお、srunを使ってcoarrayを実行する場合は、コンパイル時に **-coarray=single** と指定してください。 バッチ処理でsrunコマンドを使用せずに、プログラムを実行する場合は、 sharedやdistributedを指定して実行することも可能です。

```
## srunを使用する場合
$ ifort -coarray=single coarray.f90

## srunを使用しない場合(会話型処理を除く)
$ ifort -coarray=distributed coarray.f90
```

コンパイルしたプログラムを実行する際は、--rscオプションで pに並列数を指定すると、 その並列数でプログラムが実行されます。 -coarrayオプションにsharedやdistributedを設定した場合は、FOR_COARRAY_NUM_IMAGES という環境変数に並列数を 指定する必要があります。 また、IntelMPIをInfiniband経由で使用する場合の不具合により、**MPIR_CVAR_CH4_OFI_ENABLE_RMA** という 環境変数を「0」に設定した上で，ジョブを実行する必要があります。(2023年6月時点の情報)

```
#!/bin/bash
#============ Slurm Options ===========
#SBATCH -p gr19999b             # ジョブキュー(パーティション)の指定。投入したいキュー名に適宜変更が必要
#SBATCH -t 1:00:00              # 経過時間の指定(1時間を指定する例)。
#SBATCH --rsc p=2               # 要求リソースの指定(Coarrayを2並列で実行する場合の例)。
#SBATCH -o %x.%j.out            # ジョブの標準出力ファイルの指定。%xはジョブ名、%j はジョブIDに置換され
#============ Shell Script ============
# (任意) set -x を指定するとジョブスクリプトの実行経過を把握しやすくなります。
set -x

# IntelMPIをInfiniband経由で使用する場合の不具合対応
export MPIR_CVAR_CH4_OFI_ENABLE_RMA=0

# 並列数を環境変数に設定
export FOR_COARRAY_NUM_IMAGES=${SLURM_DPC_NPROCS}

# ジョブの実行 (-coarray=single の場合)
srun ./a.out

# 補足: -coarray=distributed や -coarray=shared の場合は、コンパイルした実行ファイル自体が
# 内部的にmpiexecを呼び出すため、srun を使用せずに実行してください。
```

コンパイル時メッセージ

Intel コンパイラは、プログラムの誤りや通知すべき情報があるときに、以下に示す形式でメッセージを出力します。

C/C++

```
ファイル名(行番号): XXX #YYY: メッセージ本文
ソースコードの該当行の内容
^
```

- XXX　：　メッセージ種別(error/warning)
- YYY　：　メッセージの通し番号
- ポインタ(^)　：　ソースコードの該当行でエラーが発見された正確な場所

出力例

```
sample.c(27): warning #175: subscript out of range
    printf(" %d , %d\n",c[1][0],c[1][10]);
                                  ^
```

Fortran

```
ファイル名（行番号）: XXX #YYY: メッセージ本文
ソースコードの該当行の内容
--------------^
```

- XXX　：　メッセージ種別（error/warning)
- YYY　：　メッセージの通し番号
- ポインタ(^)　：　ソースコードの該当行でエラーが発見された正確な場所

出力例

```
sample.f90(26): error #5560: Subscript #2 of the array C has value 20 which is greater than the upper b
print *, c(1,1),",", c(1,20)
-----------------------^
compilation aborted for sample.f90 (code 1)
```

利用可能なライブラリ

ランタイムライブラリ

Intelコンパイラは共有ストレージである /opt/system/app/intel にインストールしています。 全ノードから共通のファイルを参照することができるため、ストレージ容量の節約と管理性の向上を図ることが可能です。

一方で、全ノードからのアクセスが共有ストレージに集中する問題があります。

この問題を解決するために、 システムA/B/Cの計算ノードのローカルストレージには、intelコンパイラのランタイムライブラリ（コンパイル機能を含まないライブラリ群）をインストールしてあります。 大規模なMPIプログラムの実行、アレイジョブによる大量のプログラムの起動、短時間のプログラムを繰り返し実行する場合等にランタイムライブラリを利用すると、/opt/system の負荷を軽減し、プログラムの起動を早める効果があります。

ジョブスクリプトに次に示す**module** を切り替えるコマンドを記載いただくことで、簡易にご利用いただけますので、是非お試しください。 コンパイル時に静的リンクすることでも、同様な効果がありますが、動的リンクしかできないケースもありますので、そのような場合に是非ご利用ください。

Intelコンパイラをランタイムライブラリに切り替えるコマンド

コンパイル時と同じバージョンのランタイムライブラリに切り替えてください。 ランタイムライブラリは、バージョン番号の末尾に「-rt 」が記載されているモジュールファイルです。

module switch に -f オプションをつけて強制的にswitchする必要があります。

```
module switch -f intel/2022.3 intel/2022.3-rt -f
もしくは
module switch -f intel/2022.3-rt
```

Intel MPI をランタイムライブラリに切り替えるコマンド

Intelコンパイラと同様に、module switch に -f オプションをつけて強制的にswitchする必要があります。

```
module switch -f intelmpi/2022.3 intelmpi/2022.3-rt
もしくは
module switch -f intelmpi/2022.3-rt
```

ランタイムライブラリを使った例

以下の例は、4096並列で起動したMPIプログラムのジョブの完了までに要した経過時間を記載しています。 プログラムの演算部分を時間計測するとどちらも8秒以内に完了していますが、sacctで確認できるジョブ全体の経過時間では、ランタイムライブラリを使用することで、ジョブの経過時間が1分以上短縮されています。

```
## runtime ライブラリを未使用のジョブ(1分26秒)
$ sacct -X -j 234069 -o jobid,elapsed
JobID           Elapsed
------------ ----------
234069         00:01:26

## runtime ライブラリを使用したジョブ（17秒）
$ sacct -X -j 234060 -o jobid,elapsed
JobID           Elapsed
------------ ----------
234060         00:00:17
```

MPIライブラリ

Intel MPIライブラリが利用できます。MPIプログラムのコンパイル、リンク、実行方法については、Intel MPIライブラリをご覧ください。

数値計算ライブラリ

Intelコンパイラを利用する場合、以下の数値計算ライブラリを利用できます。各ライブラリの利用方法については、個別のページをご覧ください。

ライブラリ	システムA	システムB	システムC	システムG	クラウド
MKLライブラリ	+	+	+	-	+

+ ： すべてのユーザが利用可能
AU ： 学術研究機関限定で利用可能
- ： 利用不可

マニュアル

- Intel C++ Compiler Classic Developer Guide and Reference
- Intel Fortran Compiler Classic and Intel Fortran Compiler Developer Guide and Reference

NVIDIA HPC SDKコンパイラ

利用環境

利用できるバージョン・システム

バージョン	モジュールファイル名	システムA	システムB/C	システムG	クラウドシステム	備考
23.9 (default)	nvhpc/23.9	-	-	+	-	2024年4月導入
23.5	nvhpc/23.5	-	-	+	-	2023年8月導入
22.9	nvhpc/22.9	-	-	+	-	2023年1月導入

+ ： すべてのユーザが利用可能
- ： 利用不可

システムGでは、システムにログインした時点で、NVIDIA HPC SDKコンパイラがデフォルトで設定されています。システムB、C、クラウドではNVIDIA HPC SDKコンパイラは使えません。

ログイン時に自動で環境設定を行いたい場合は、ログインシェルの起動ファイルに必要なmoduleコマンドを記述してください。詳細は 環境設定 をご覧ください。 moduleコマンドの詳細は Modules をご覧ください。

cudaについて

バージョン	モジュールファイル名	システムA	システムB/C	システムG	クラウドシステム	備考
12.2.2 (dafault)	cuda/12.2.2	-	-	+	-	2024年4月導入
12.1.1	cuda/12.1.1	-	-	+	-	2023年8月導入
11.7.1	cuda/11.7.1	-	-	+	-	2023年1月導入
9.2	cuda/9.2	-	-	+	-	2023年1月導入

+ ： すべてのユーザが利用可能
- ： 利用不可

NVIDIA HPC SDKコンパイラのモジュールファイルをダウンロードするだけでcudaもロードされます。 cudaのデフォルトバージョンは11.7ですが、バージョンを切り替えたい場合は、PrgEnvNvidia がロードされている状態で、以下のようにmoduleコマンドを実行してください。

```
$ module load cuda/9.2
```

コンパイル方法

コマンド

言語	コマンド	実行形式
Fortran	nvfortran	nvfortran [オプション] ファイル名
C	nvc	nvc [オプション] ファイル名
C++	nvc++	nvc++ [オプション] ファイル名

オプション

- 主要オプション（並列化、最適化など）

オプション名	説明
-o FILENAME	オブジェクトファイルの名前を指定します。
-mcmodel=medium	2Gbyteを超えるメモリを使用できるようになります。
-mp	OpenMP指示子を有効にしてコンパイルします。
-Mconcur	自動並列化を行います。
-O0/-O1/-O2/-O3/-O4	最適化のレベルを指定します
-fast	一般的な最適化機能を有効にします。
-Mipa=fast	手続き間の解析処理を最適化します。(注意事項) 現在のバージョンでは使用できなくなりました。

- メッセージ出力とデバッグのオプション

オプション名	説明
-Mlist	リスティングファイルを作成します。
-Minform=inform	すべてのエラーメッセージを表示します。
-Minfo[=OPTION]	標準エラー出力に情報を表示します。 表示する情報はオプションで指定します。
-Mneginfo[=OPTION]	実施されなかった最適化についての情報を表示します。

- Fortran言語固有オプション

オプション名	説明
-Mfixed	プログラムが固定形式で記述されていることを指示します。
-Mfree	プログラムが自由形式で記述されていることを指示します。

オプション名	説明
-Mstandard	ANSI規格準拠でない構文を警告します。
-Mdclchk	暗黙の型宣言を警告します。
-C	プログラム実行時に、配列の領域外参照を検出します。

- GPU関連のオプション

オプション名	説明
-gpu	GPUに関する設定を指定します。-acc、-cuda、-mp、-stdparオプションと併せて使用します。 （例）-gpu=cc80 -acc ※ccはcompute capabilityを意味します。システムGに搭載のGPUはA100ですので、ccを指定する場合は、cc80としてください。
-acc	OpenACCを有効にします。
-cuda	CUDAを有効にします。
-Minfo=accel	GPU関連のコンパイル情報を出力します。

コンパイル例

逐次プログラム

```
$ nvfortran test.f90    # Fortranの例
$ nvc test.c       # C言語の例
$ nvc++ test.cpp     # C++の例
$ tssrun ./a.out    # 実行
```

自動並列化の利用

```
$ nvfortran -Mconcur test.f90
$ tssrun --rsc p=1:t=4:c=4 ./a.out # 並列数4を指定して実行
```

OpenMPの利用

OpenMPは、プログラムの並列化のためのオープン規格です。 ソースコードに#pragma omp(C/C++)もしくは!$ompで始まる指示を書き込み、所定のオプションをつけてコンパイルするだけで、コンパイラに自動で並列化を行わせることができます。

OpenMPへの指示を書き込んだソースコードをコンパイルするには、-mpオプションをつけます。

```
$ nvc -mp test.c
```

コンパイルしたプログラムを実行する時、--rscオプションでtとcに並列数を指定すると、その並列数でプログラムが実行されます。

```
$ tssrun --rsc p=1:t=8:c=8 ./a.out # 並列数8を指定して実行
```

OpenACCの利用

OpenACCは、プログラムの並列化のためのオープン規格です。 ソースコードに#pragma acc(C/C++)もしくは!$accで始まる指示を書き込み、所定のオプションをつけてコンパイルするだけで、GPU上で動作する実行コードを生成します。

OpenACCへの指示を書き込んだソースコードをコンパイルするには、-accオプションをつけます。

```
$ nvfortran -acc test.f90
```

利用可能なライブラリ

MPIライブラリ

OpenMPIライブラリが利用できます。MPIプログラムのコンパイル、リンク、実行方法については、 OpenMPIライブラリ をご覧ください。

マニュアル

Version 22.9

- NVIDIA HPC Compilers User's Guide
- NVIDIA HPC Compilers Reference Guide

GNUコンパイラ

利用環境

利用できるバージョン・システム

バージョン	モジュールファイル名	システムA	システムB/C	システムG	クラウドシステム	備考
12.2.0	gcc/12.2.0	+	+	+	+	2022年11月導入
8.5.0	-	+	+	+	+	OS標準

+ ： すべてのユーザが利用可能
- ： 利用不可

システムにログインした時点では、Intelコンパイラもしくは NVIDIA HPC SDKコンパイラがデフォルトで利用できるようになっています。以下のようにmoduleコマンドを実行し、コンパイラを切り替えてください。

```
（システムA、B、C、クラウド）
$ module switch PrgEnvIntel PrgEnvGCC
（システムG）
$ module switch PrgEnvNvidia PrgEnvGCC
```

システムにログインした時点で、バージョン8.5.0が使えます。バージョン12.2.0を使いたい場合は、PrgEnvGCC がロードされている状態で、以下のようにmoduleコマンドを実行してください。

```
$ module load gcc/12.2.0
```

コンパイル方法

コマンド

言語	コマンド	実行形式
Fortran	gfortran	gfortran オプション ファイル名
C	gcc	gcc オプション ファイル名
C++	g++	g++ オプション ファイル名

オプション

- 主要オプション（並列化、最適化など）

オプション名	説明
-o *FILENAME*	オブジェクトファイルの名前を指定します。
-mcmodel=medium	2Gbyteを超えるメモリをサポートします。
-fopenmp	OpenMP指示子を有効にしてコンパイルします。
-O0/-O1/-O2/-O3	最適化のレベルを指定します（デフォルトは-O0）。

- メッセージ出力とデバッグのオプション

オプション名	説明
-Wall	すべての警告メッセージを表示します。

- **Fortran**言語固有オプション

オプション名	説明
-ffixed-form	プログラムが固定形式で記述されていることを指示します。
-ffree-form	プログラムが自由形式で記述されていることを指示します。
-pedantic	Fortran拡張の利用を警告します。
-fimplicit-none	暗黙の型宣言を警告します。

コンパイル例

逐次プログラム

```
$ gfortran test.f90    # Fortranの例
$ gcc test.c           # C言語の例
$ g++ test.cpp         # C++の例
$ tssrun ./a.out       # 実行
```

OpenMPの利用

OpenMPは、プログラムの並列化のためのオープン規格です。 ソースコードに#pragma ompで始まる指示を書き込み、所定のオプションをつけてコンパイルするだけで、コンパイラに自動で並列化を行わせることができます。

OpenMPへの指示を書き込んだソースコードをコンパイルするには、-fopenmpオプションをつけます。

```
$ gcc -fopenmp test.c
```

コンパイルしたプログラムを実行する時、-Aオプションでtとcに並列数を指定すると、その並列数でプログラムが実行されます。

```
$ tssrun --rsc p=1:t=8:c=8 ./a.out # 並列数8を指定して実行
```

利用可能なライブラリ

MPIライブラリ

IntelMPIライブラリ、OpenMPIライブラリが利用できます。MPIプログラムのコンパイル、リンク、実行方法については、IntelMPIライブラリ、 OpenMPIライブラリ をご覧ください。

マニュアル

- GCC online documentation

リンク

- GCC, the GNU Compiler Collection

Intel MPIライブラリ

利用環境

利用できるバージョン・システム

バージョン	モジュールファイル名	システムA	システムB/C	システムG	クラウドシステム	備考
2024.0	intel/2024.0	+	+	-	+	2024年4月導入. Intelコンパイラ + MKL + TBB
2023.2 (default)	intel/2023.2	+	+	-	+	2024年4月導入. Intelコンパイラ + MKL + TBB
2023.2-rt	intel/2023.2-rt	+	+	-	+	2024年4月導入. ランタイムライブラリ
2023.1	intel/2023.1	+	+	-	+	2023年6月導入.
2023.1-rt	intel/2023.1-rt	+	+	-	+	2023年6月導入. ランタイムライブラリ
2022.3	intelmpi/2022.3	+	+	-	+	2022年11月導入
2022.3-rt	intelmpi/2022.3-rt	+	+	-	+	2022年11月導入. ランタイムライブラリ

注：Intel MPIの個別のバージョンではなく、Intel OneAPI の包括的なバージョンで記載しています。

+ ： すべてのユーザが利用可能
- ： 利用不可

スレッド並列サポートレベル

Intel MPIで提供可能なスレッド並列サポートレベルは、MPI_THREAD_MULTIPLEです。各スレッドから自由にMPI関数を呼び出すことができます。

コンパイル・実行方法

コンパイル

コンパイルコマンド（**Intel**コンパイラ）

言語	コマンド	実行形式
Fortran	mpiifort	mpiifort オプション ファイル名

言語	コマンド	実行形式
C	mpiicc	mpiicc オプション ファイル名
C++	mpiicpc	mpiicpc オプション ファイル名

コンパイルコマンド（**GNU**コンパイラ）

言語	コマンド	実行形式
Fortran 95	mpif90	mpif90 オプション ファイル名
Fortran 77	mpif77	mpif77 オプション ファイル名
C	mpigcc	mpicc オプション ファイル名
C++	mpigcxx	mpicxx オプション ファイル名

コンパイル例

```
$ mpiifort -O3 -parallel sample_mpi.f90
```

実行方法

コンパイルしたMPIプログラムを実行するためには、 会話型処理またはバッチ処理を利用し、計算ノード上でプログラムを起動する必要があります。 詳細は、 プログラム実行例 をご覧ください。

プログラム実行例

会話型処理の実行例

会話型処理は **tssrun** コマンドを使用することで簡易に実行可能です。--rsc オプションにより、確保するリソース量を調整することが可能です。 tssrunコマンドの詳細については、 会話型処理 のページをご覧ください。

MPIプログラムを**8**プロセスで実行する場合

```
$ tssrun --rsc p=8 ./a.out
```

ハイブリッド並列の場合（**MPI 4**プロセス、**OpenMP 8**スレッド）

```
$ tssrun --rsc p=4:t=8:c=8 ./a.out
```

バッチ処理の実行例

バッチ処理は、Slurmのオプションと処理内容を記載したジョブスクリプトを用意し、sbatchコマンドで処理を依頼する必要があります。 以下に例を示しますが、オプション等の詳細については、 バッチ処理 のページをご覧ください。

MPIプログラムを8プロセスで実行する場合

```
$ cat sample.sh
#!/bin/bash
#============ SBATCH Directives =======
#SBATCH -p gr19999b
#SBATCH -t 1:0:0
#SBATCH --rsc p=8
#SBATCH -o %x.%A.out

#============ Shell Script ============
srun ./a.out
$ sbatch sample.sh
```

ハイブリッド並列の場合（**MPI 4プロセス、OpenMP 8スレッド**）

```
$ cat sample.sh
#!/bin/bash
#============ SBATCH Directives =======
#SBATCH -p gr19999b
#SBATCH -t 1:0:0
#SBATCH --rsc p=4:t=8:c=8:m=8G
#SBATCH -o %x.%A.out

#============ Shell Script ============
srun ./a.out
$ sbatch sample.sh
```

マニュアル

- Intel MPI Library Developer Guide for Linux* OS Developer Guide
- Intel MPI Library Developer Guide for Linux* OS Developer Reference

リンク

- Intel MPI Library

OpenMPIライブラリ

利用環境

利用できるバージョン・システム

バージョン	モジュールファイル名	システム A	システム B/C	システム G	クラウドシステム	備考
4.1.5	openmpi/4.1.5_nvhpc-23.9	-	-	+	-	2024年4月導入
4.1.5	openmpi/4.1.5_nvhpc-23.5	-	-	+	-	2023年8月導入
4.0.5	openmpi/4.0.5_intel-2022.3	+	+	-	+	2023年6月導入
4.0.5	openmpi/4.0.5_gnu-8.5.0	+	+	+	+	2022年11月導入
4.0.5	openmpi/4.0.5_nvhpc-22.9	-	-	+	-	2022年11月導入
4.0.5	openmpi/4.0.5_nvhpc-22.5	-	-	+	-	2022年11月導入

+ ： すべてのユーザが利用可能
- ： 利用不可

スレッド並列サポートレベル

OpenMPIで提供可能なスレッド並列サポートレベルは、MPI_THREAD_MULTIPLEです。各スレッドから自由にMPI関数を呼び出すことができます。

コンパイル・実行方法

コンパイル

コンパイルコマンド

言語	コマンド	実行形式
Fortran	mpifort	mpifort オプション ファイル名
C	mpicc	mpicc オプション ファイル名
C++	mpic++	mpic++ オプション ファイル名

コンパイル例

```
$ mpifort -O3 sample_mpi.f90
```

実行

コンパイルしたMPIプログラムを実行するためには、 会話型処理では **tssrun** コマンドを、バッチ処理では **srun** コマンドを使用します。会話型実行、バッチ実行ともに、並列数は --rsc オプションの引数 p で指定します。詳細は、 サンプル をご覧ください。

実行例（会話型）

```
$ tssrun --rsc p=2 ./a.out
```

サンプル

会話型でのMPIプログラム実行

会話型実行の詳細については、 会話型処理 のページをご覧ください。

- **8**並列で実行

```
$ tssrun --rsc p=8 ./a.out
```

- スレッド並列と組み合わせて実行（**MPI 4**並列、**OpenMP 8**並列）

```
$ tssrun --rsc p=4:t=8:c=8 ./a.out
```

バッチでのMPIプログラム実行

バッチ実行の詳細については、 バッチ処理 のページをご覧ください。

- **8**並列で実行

```
$ cat sample.sh
#!/bin/bash
#============= SBATCH Directives =======
#SBATCH -p gr19999b
#SBATCH -t 1:0:0
#SBATCH --rsc p=8
#SBATCH -o %x.%A.out

#============= Shell Script ============
srun ./a.out
$ sbatch sample.sh
```

- スレッド並列と組み合わせて実行（**MPI 4**並列、**OpenMP 8**並列）

```
$ cat sample.sh
#!/bin/bash
#============= SBATCH Directives =======
#SBATCH -p gr19999b
#SBATCH -t 1:0:0
#SBATCH --rsc p=4:t=8:c=8:m=8G
#SBATCH -o %x.%A.out

#============= Shell Script ============
srun  ./a.out
$ sbatch sample.sh
```

マニュアル

- Open MPI v4.0 documentation

リンク

- Open MPI: Open Source High Performance Computing

Intel VTune Profiler

利用環境

利用できるバージョン・システム

バージョン	モジュールファイル名	システムA	システムB/C	システムG	クラウドシステム	備考
2024.0	intel-vtune/2024.0	+	+	-	-	
2023.2 (default)	intel-vtune/2023.2	+	+	-	-	
2023.2	intel-vtune/2023.1	+	+	-	-	
2022.3	intel-vtune/2022.3	+	+	-	-	

+ : すべてのユーザが利用可能
- : 利用不可

Intelコンパイラ が利用できる状態で、以下のようにmoduleコマンドを実行してください。

```
$ module load intel-vtune
```

moduleコマンドの詳細は Modules をご覧ください。

利用方法

コマンド

コマンド	説明
vtune-gui	GUI版のVTune Profilerを起動します。
vtune	コマンドライン版のVTune Profilerを起動します。

オプション

vtune の主なオプション

オプション	説明
-collect=*string*	分析を行うタイプを指定します。
-app-working-dir=*string*	ワーキングディレクトリを指定します。

オプション	説明
-r, -result-dir=*string*	結果を保存するディレクトリを指定します。

vtuneの-collectで指定可能な主なタイプ

タイプ	説明
threading	マルチスレッドの並列性を表示します。
hotspots	ホットスポットを分析します。
memory-access	メモリアクセスを分析します。

利用例

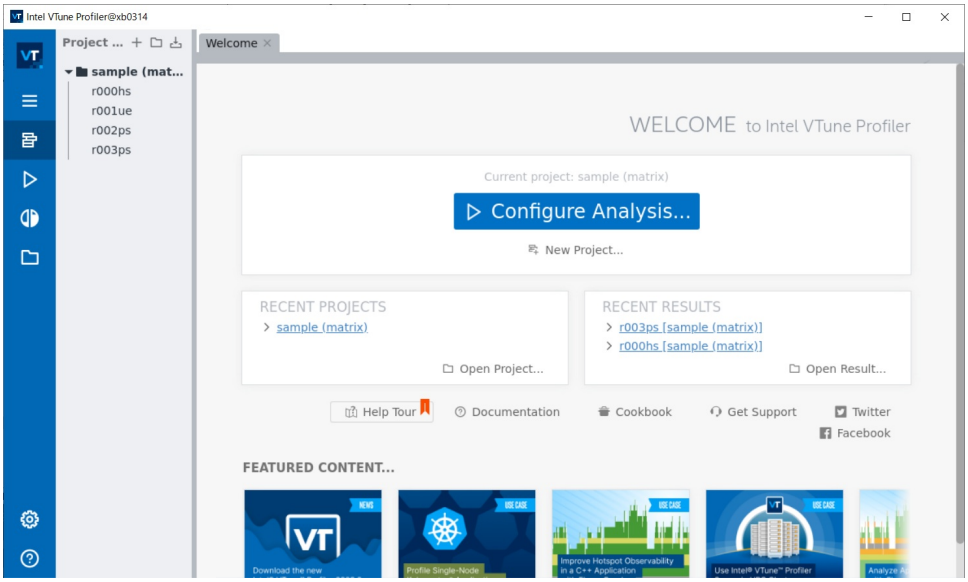
GUIでの利用例

1. コンパイル
   Intel VTune Profilerを使用する場合、デバッグオプション **-g** と実際のプログラム実行時と同等な最適化オプションを指定してコンパイルを行ってください。

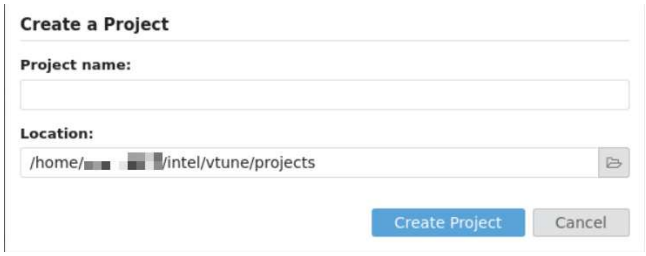
```
$ icc -g -O2 test.c
```

2. VTune Profiler の起動
   MobaXterm または FastX で、 **vtune-gui** コマンドを実行するとVTune Amplifierが起動します。 **tssrun** コマンドの詳細は 会話型処理 をご覧ください。

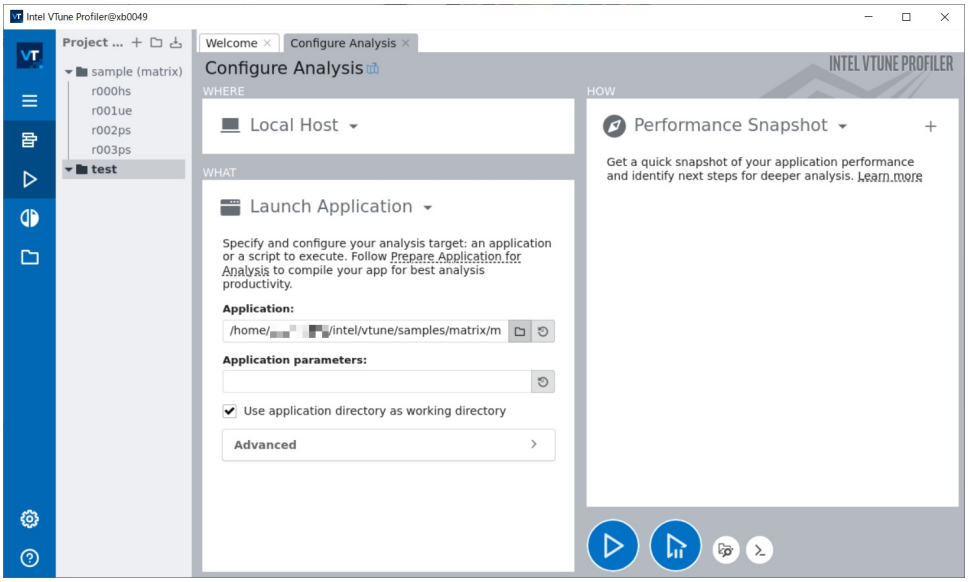
```
$ tssrun --x11 vtune-gui
```



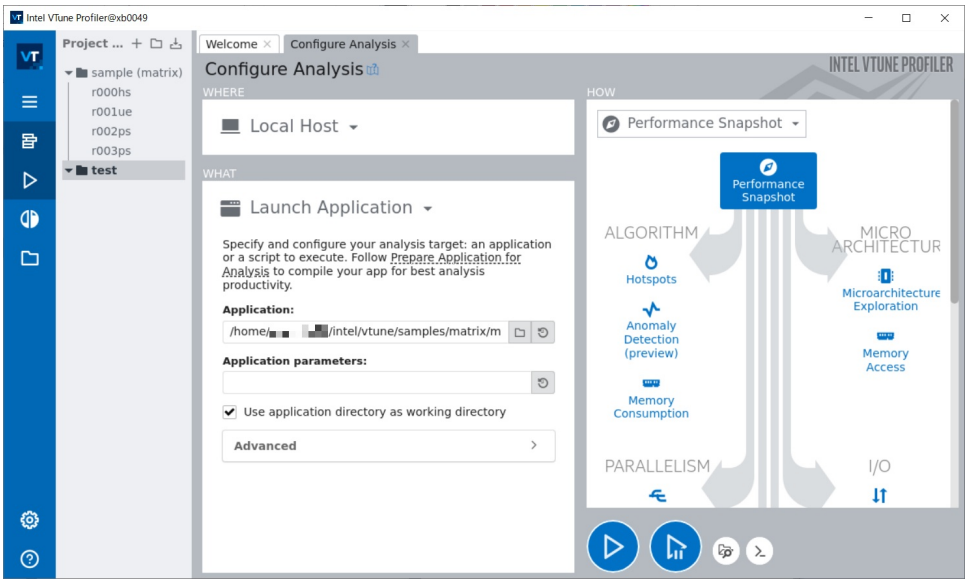
3. プロジェクトを作成する
   画面中央の **New Project** を選択すると以下のような画面が表示されますので、適切な **Project name** を入力して、**Create Project** を押してください。



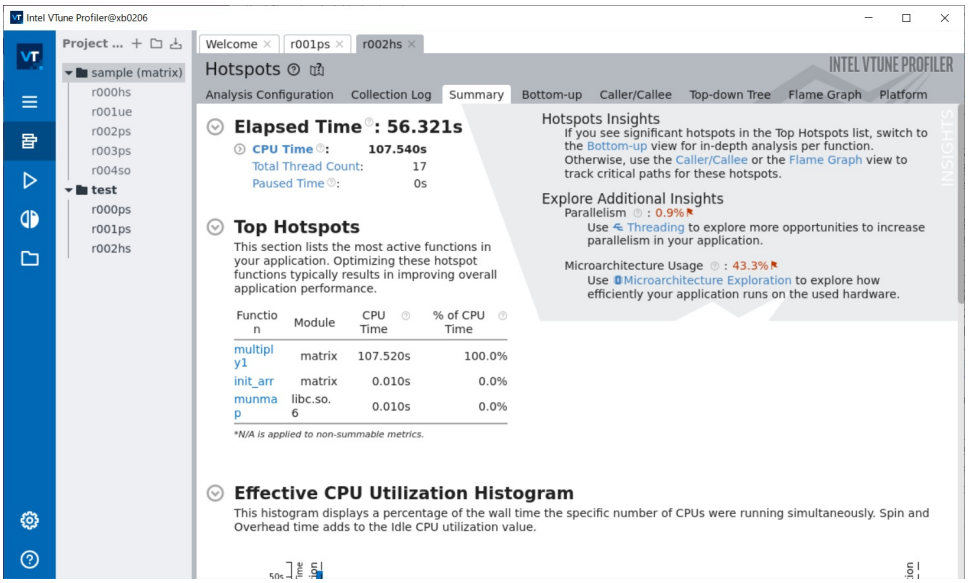
4. プロジェクトの設定
   続いて、Configure Analysis の画面が表示されますので、対象とするプログラムを **Application** で指定して **OK** を押してください。引数等が必要な場合はここで指定します。



5. 分析開始
   Performance Snapshot を押して、チェックの対象を指定したうえで、三角マークの **Start** を押してください。



6. 分析結果の確認
   しばらく待つと、VTune Profilerでの分析結果が表示されます。この例では、multiply1関数がCPU時間を多く使用していることが分かります。



OpenMPプログラムの解析

GUI版はOpenMPプログラムの解析に対応しています。 tssrun コマンドで起動時に、--rsc オプションでスレッド数を指定します。

例：8スレッド実行

```
tssrun --x11 --rsc t=8:c=8 vtune-gui
```

MPIプログラムの解析

MPIプログラムの解析は、CUI版をご利用ください。

CUIでの利用例

1. コンパイル
   Intel VTune Profilerを使用する場合、デバッグオプション **-g** と実際のプログラム実行時と同等な最適化オプションを指定してコンパイルを行ってください。

```
$ icc -g -O2 test.c
```

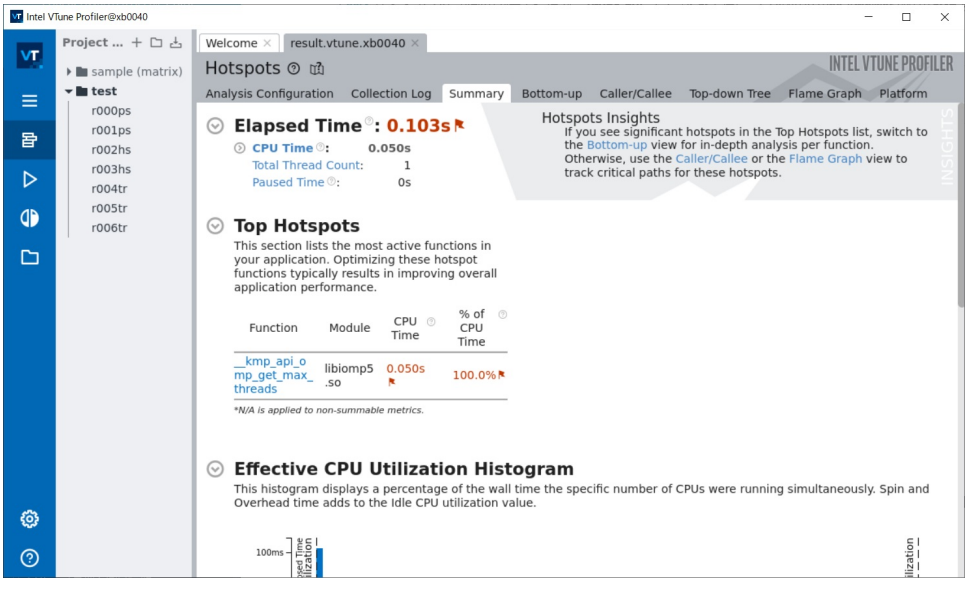
2. チェックの実行
   CUI版のVTune Profilerは **vtune** コマンドで利用できます。この例では、結果を出力するディレクトリの指定して、hotspotsの分析をしています。 **tssrun** コマンドの詳細は 会話型処理 をご覧ください。

```
$ tssrun vtune -collect hotspots  -r=./result.vtune ./a.out
salloc: Pending job allocation 723318
salloc: job 723318 queued and waiting for resources
salloc: job 723318 has been allocated resources
salloc: Granted job allocation 723318
salloc: Waiting for resource configuration
salloc: Nodes xb0013 are ready for job
vtune: Analyzing data in the node-wide mode. The hostname (xb0013) will be added to the result path/nam
vtune: Collection started.

 (略)

vtune: Executing actions 42 % Saving the resultElapsed Time: 0.075s
 | Application execution time is too short. Metrics data may be unreliable.
 | Consider reducing the sampling interval or increasing your application
 | execution time.
 |
    CPU Time: 0.050s
        Effective Time: 0s
        Spin Time: 0s
            Imbalance or Serial Spinning: 0s
            Lock Contention: 0s
            Other: 0s
        Overhead Time: 0.050s
         | A significant portion of CPU time is spent in synchronization or
         | threading overhead. Consider increasing task granularity or the scope
         | of data synchronization.
         |
            Creation: 0s
            Scheduling: 0s
            Reduction: 0s
            Atomics: 0s
            Other: 0.050s
    Total Thread Count: 1
    Paused Time: 0s

Top Hotspots
Function                       Module       CPU Time  % of CPU Time(%)
-----------------------------  -----------  --------  ----------------
__kmp_api_omp_get_max_threads  libiomp5.so    0.050s            100.0%
Effective Physical Core Utilization: 100.0% (112.000 out of 112)
```

```
    Effective Logical Core Utilization: 148.8% (333.230 out of 224)
Collection and Platform Info
    Application Command Line: ../oss/openmp/omp-sample/01_hello/src/c/run.x
    Operating System: 4.18.0-477.15.1.el8_8.x86_64 Red Hat Enterprise Linux release 8.8 (Ootpa)
    Computer Name: xb0013
    Result Size: 5.5 MB
    Collection start time: 09:39:56 21/09/2023 UTC
    Collection stop time: 09:39:57 21/09/2023 UTC
    Collector Type: Event-based counting driver,User-mode sampling and tracing
    CPU
        Name: Intel(R) Xeon(R) Processor code named Sapphirerapids
        Frequency: 2.000 GHz
        Logical CPU Count: 224
        LLC size: 110.1 MB
        Cache Allocation Technology
            Level 2 capability: available
            Level 3 capability: available

If you want to skip descriptions of detected performance issues in the report,
enter: vtune -report summary -report-knob show-issues=false -r <my_result_dir>.
Alternatively, you may view the report in the csv format: vtune -report
<report_name> -format=csv.
vtune: Executing actions 100 % done
salloc: Relinquishing job allocation 723318

exit code: 0
```

3. 結果の確認
   MobaXterm または FastX など X（GUI）が利用できる環境で、 **vtune-gui** コマンドを実行し、GUI版のVTune Amplifierで結果を確認してください。 **tssrun** コマンドの詳細は 会話型処理 をご覧ください。

```
$ tssrun --x11 vtune-gui ./result/result.vtune
```



並列プログラムの解析

CUI版はMPIおよびOpenMPプログラムの解析に対応しています。 tssrunでVTuneを実行の際、--rsc オプションで並列数を指定します。

例：MPI4並列で解析

```
$ tssrun --rsc p=4 vtune -collect hotspots  -r=./result ./a.out
salloc: Pending job allocation 723080
salloc: job 723080 queued and waiting for resources
salloc: job 723080 has been allocated resources
salloc: Granted job allocation 723080
vtune: Analyzing data in the node-wide mode. The hostname (xb0127) will be added to the result path/nam
vtune: Collection started.

 (略)

vtune: ExecutinElapsed Time: 1.015sg the result
    CPU Time: 1.710s
        Effective Time: 1.710s
        Spin Time: 0s
            MPI Busy Wait Time: 0s
            Other: 0s
        Overhead Time: 0s
            Other: 0s
    Total Thread Count: 8
    Paused Time: 0s

Top Hotspots
Function                Module                CPU Time  % of CPU Time(%)
----------------------  --------------------  --------  ----------------
read                    libc.so.6               0.670s             39.2%
PMPI_Init               libmpi.so.12            0.112s              6.6%
main                    allrank                 0.100s              5.8%
dlopen                  libdl.so.2              0.089s              5.2%
[ld-linux-x86-64.so.2]  ld-linux-x86-64.so.2    0.080s              4.7%
[Others]                N/A                     0.658s             38.5%
Effective Physical Core Utilization: 89.5% (100.261 out of 112)
    Effective Logical Core Utilization: 45.0% (100.843 out of 224)
     | The metric value is low, which may signal a poor utilization of logical
     | CPU cores while the utilization of physical cores is acceptable. Consider
     | using logical cores, which in some cases can improve processor throughput
     | and overall performance of multi-threaded applications.
     |
Collection and Platform Info
    Application Command Line: ../lecture/20230906/mpi/allrank
    Operating System: 4.18.0-477.15.1.el8_8.x86_64 Red Hat Enterprise Linux release 8.8 (Ootpa)
    Computer Name: xb0127
    Result Size: 7.2 MB
    Collection start time: 09:24:37 21/09/2023 UTC
    Collection stop time: 09:24:38 21/09/2023 UTC
    Collector Type: Event-based counting driver,User-mode sampling and tracing
    CPU
        Name: Intel(R) Xeon(R) Processor code named Sapphirerapids
        Frequency: 2.000 GHz
        Logical CPU Count: 224
        LLC size: 110.1 MB
        Cache Allocation Technology
```

```
            Level 2 capability: available
            Level 3 capability: available

If you want to skip descriptions of detected performance issues in the report,
enter: vtune -report summary -report-knob show-issues=false -r <my_result_dir>.
Alternatively, you may view the report in the csv format: vtune -report
<report_name> -format=csv.
vtune: Executing actions 100 % done
salloc: Relinquishing job allocation 723080

exit code: 0
```

マニュアル

- Intel VTune Profiler Documentation
- Intel VTune Progiler User Guide

Intel oneAPI DPC++/C++コンパイラ

利用環境

利用できるバージョン・システム

バージョン	システムA	システムB	システムC	システムG	クラウド	備考
2022.1.2	+	+	+	-	+	Intel oneAPI 2022

環境設定

バージョン	moduleファイル名
2022.1.2 (default)	intel/2022

システムにログインした時点で、Intelコンパイラがデフォルトで設定されています。

```
$ module list
Currently Loaded Modulefiles:
x) SysB/2022   x) PrgEnvIntel/2022  x) intel/2022
```

ログイン時に自動で環境設定を行いたい場合は、ログインシェルの起動ファイルに必要なmoduleコマンドを記述してください。詳細は環境設定をご覧ください。 moduleコマンドの詳細はModulesをご覧ください。

コンパイル方法

コマンド

言語	コマンド	実行形式
DPC++	dpcpp	dpcpp [オプション] ファイル名
C	icx	icx [オプション] ファイル名
C++	icpx	icpx [オプション] ファイル名

オプション

オプション名	説明
-o FILENAME	オブジェクトファイルの名前を指定します。

オプション名	説明
-mcmodel=medium	2Gbyteを超えてメモリを使用できるようになります。 ※SYCLでは使用できません。
-shared-intel	インテルが提供するライブラリをすべて動的にリンクします。
-fpic	位置に依存しないコードを生成します。
-qopenmp	OpenMP指示子を有効にしてコンパイルします。 指定した GPU へのオフロードを有効にするには、-fopenmp-targets を併せて使用します。
-O0/-O1/-O2/-O3	最適化のレベルを指定します(デフォルトは-O2)
-fast	プログラムの実行速度が最大になるように最適化します。-fastオプションにより、次のオプションが付与されます。 -ipo, -O3, -static, -fp-model fast=2
-ipo	複数ファイル間で、手続き間の処理を最適化します。
-qopt-report	実施した最適化についての情報をYAMLで出力します。
-qopt-report-file=KEYWORD	実施した最適化についての情報を、ファイル(filename) / 標準エラー出力(stderr) / 標準出力(stdout)に出力します。

コンパイル例

逐次プログラム

```
$ dpcpp test.cpp  # dpc++の例
$ icx test.c          # C言語の例
$ icpx test.cpp     # C++の例
$ tssrun ./a.out    # プログラムの実行
```

OpenMPの利用

OpenMPは、プログラムの並列化のためのオープン規格です。 ソースコードに#pragma ompで始まる指示を書き込み、所定のオプションをつけてコンパイルするだけで、コンパイラに自動で並列化を行わせることができます。

OpenMPへの指示を書き込んだソースコードをコンパイルするには、-qopenmpオプションをつけます。

```
$ icx -qopenmp test.c
```

コンパイルしたプログラムを実行する際、-Aオプションでtとcに並列数を指定すると、その並列数でプログラムが実行されます。

```
$ tssrun -A p=1:t=8:c=8 ./a.out # 並列数8を指定して実行
```

マニュアル

Intel Fortran コンパイラ & Intel Fortran コンパイラ クラシック

利用環境

利用できるバージョン・システム

バージョン	システムA	システムB	システムC	システムG	クラウド	備考
2022.1.2	+	+	+	-	+	Intel oneAPI 2022

環境設定

バージョン	moduleファイル名
2022.1.2 (default)	intel/2022

システムにログインした時点で、Intelコンパイラがデフォルトで設定されています。

```
$ module list
Currently Loaded Modulefiles:
x) SysB/2022   x) PrgEnvIntel/2022  x) intel/2022
```

ログイン時に自動で環境設定を行いたい場合は、ログインシェルの起動ファイルに必要なmoduleコマンドを記述してください。詳細は環境設定をご覧ください。 moduleコマンドの詳細はModulesをご覧ください。

コンパイル方法

コマンド

コンパイラ	コマンド	実行形式
Intel Fortran コンパイラ クラシック	ifort	ifort [オプション] ファイル名
Intel Fortran コンパイラ	ifx	ifx [オプション] ファイル名

主なオプション

オプション名	説明
-o FILENAME	オブジェクトファイルの名前を指定します。

オプション名	説明
-mcmodel=medium	2Gbyteを超えてメモリを利用できるようになります。
-shared-intel	インテルが提供するライブラリをすべて動的にリンクします。
-fpic	位置に依存しないコードを生成します。
-qopenmp	OpenMP指示子を有効にしてコンパイルします。 指定した GPU へのオフロードを有効にするには、-fopenmp-targets (*1) を併せて使用します。
-parallel (*2)	自動並列化を行います。
-O0/-O1/-O2/-O3	最適化のレベルを指定します（デフォルトは-O2）。
-fast	プログラムの実行速度が最大になるように最適化します。-fast オプションにより、次のオプションが付与されます。 `-ipo, -O3, -no-prec-div,-static, -fp-model fast=2, and -xHost`
-ip (*2)	単一ファイル内で、手続き間の処理を最適化します。
-ipo	複数ファイル間で、手続き間の処理を最適化します。
-qopt-report	実施したすべての最適化についての情報を表示します。(*2) 実施した最適化についての情報をYAMLで出力します。(*1)
-qopt-report-file=KEYWORD	実施した最適化についての情報を、ファイル(filename) / 標準エラー出力(stderr) / 標準出力(stdout)に出力します。このオプションを使用する場合、オプション -qopt-report を指定する必要はありません。
-free/-nofixed	プログラムが自由形式で記述されていることを明示します。
-nofree/-fixed	プログラムが固定形式で記述されていることを明示します。
-warn all	すべての警告メッセージを表示します。declarations,alignments等を指定することも可能です。
-check all	すべての実行時診断機能を有効にします。bounds,uninit等を指定することも可能です。

(*1) ifx でのみ使用可
(*2) ifort でのみ使用可

コンパイル例

逐次プログラム

```
$ ifort test.f90   # ifortの例
$ ifx test.f90     # ifxの例
```

自動並列化の利用

```
$ ifort -parallel test.f90
$ tssrun -A p=1:t=4:c=4 ./a.out # 並列数4を指定して実行
```

※ifxでは使用不可

OpenMPの利用

OpenMPは、プログラムの並列化のためのオープン規格です。 ソースコードに#pragma ompで始まる指示を書き込み、所定のオプションをつけてコンパイルするだけで、コンパイラに自動で並列化を行わせることができます。

OpenMPへの指示を書き込んだソースコードをコンパイルするには、-qopenmpオプションをつけます。

```
$ icx -qopenmp test.c
```

コンパイルしたプログラムを実行する際、-Aオプションでtとcに並列数を指定すると、その並列数でプログラムが実行されます。

```
$ tssrun -A p=1:t=8:c=8 ./a.out # 並列数8を指定して実行
```

コンパイル時メッセージ

Intel Fortran コンパイラは、プログラムの誤りや通知すべき情報があるときに、以下に示す形式でメッセージを出力します。

```
ファイル名（行番号）: XXX #YYY: メッセージ本文
ソースコードの該当行の内容
--------------^
```

- XXX　：　メッセージ種別（error/warning)
- YYY　：　メッセージの通し番号
- ポインタ(---^)　：　ソースコードの該当行でエラーが発見された正確な場所

出力例

```
sample.f90(26): error #5560: Subscript #2 of the array C has value 20 which is greater than the upper b
print *, c(1,1),",", c(1,20)
-----------------------^
compilation aborted for sample.f90 (code 1)
```

利用可能なライブラリ

MPIライブラリ

数値計算ライブラリ

マニュアル

Intel Trace Analyzer / Collector

利用環境

利用できるバージョン・システム

バージョン	システムA	システムB	システムC
Trace Analyzer 2022 (default)	—	+	+
Trace Analyzer 2021	—	+	+

+ ： 利用可能
— ： 利用不可

環境設定

バージョン	moduleファイル名
2022	iutils/xe2022
2021	iutils/xe2021

Intelコンパイラ が利用できる状態で、以下のようにmoduleコマンドを実行してください。

```
$ module load iutils
```

moduleコマンドの詳細は Modules をご覧ください。

Intel Trace Analyzer / Collector を使用してプログラムを実行すると、計算ノードのテンポラリ領域（/tmp）に生成されたデータが出力されます。計算ノードの /tmp の容量には限りがあるため、データが溢れてしまい、プログラムの実行結果に影響が出る場合があります。データの出力先を、環境変数TMPDIRで指定したディレクトリに変更できますので、LARGE領域などを出力先として利用してください。

（設定例）

```
（tcsh の場合）
$ setenv TMPDIR /LARGE0/gr19999/b59999/output
（bash の場合）
$ export TMPDIR=/LARGE0/gr19999/b59999/output
```

利用方法

コマンド

コマンド	説明
traceanalyzer	Trace Analyzerを起動します。

利用例

コンパイル時にトレースファイルの取得を指示する方法

1. コンパイル
   Intel Trace Analyzer / Collectorを使用する場合、 **-trace** オプションを指定してコンパイルを行ってください。

```
$ mpiicc  test.c -trace
```

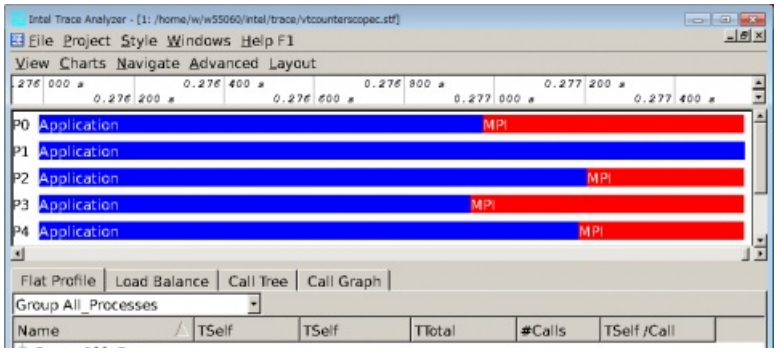
2. 実行
   -trace オプションを指定してコンパイルしたプログラムを実行すると、トレース情報が記録された **stf**ファイル が作成されます。以下の例では、tssrun コマンドを使用して、会話型で実行しています。

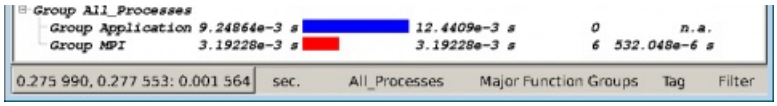
```
$ tssrun -A p=4 mpiexec.hydra ./a.out
Job <582271> is submitted to queue <tb>.
<<Waiting for dispatch ...>>
<<Starting on gb-0004>>
[0] Intel(R) Trace Collector INFO: Writing tracefile a.out.stf in /home/b/b59999/
```

3. Intel Trace Analyzer / Collector でのトレース結果確認
   Exceed onDemand など X（GUI）が利用できる環境で、出力された **stf**ファイル を指定して **traceanalyzer** コマンドを実行するとIntel Trace Analyzerが起動します。

```
$ traceanalyzer a.out.stf
```

下図は、 **Charts** => **Event Timeline** でタイムライン表示を行った例です。





実行時にトレースファイルの取得を指示する方法

1. コンパイル
   通常と同じ手順でコンパイルを行ってください。

```
$ mpiicc  test.c
```

2. 実行
   -trace オプションを指定して実行すると、トレース情報が記録された **stf**ファイル が作成されます。以下の例では、tssrun コマンドを使用して、会話型で実行しています。

```
$ tssrun -A p=4 mpiexec.hydra -trace ./a.out
Job <582271> is submitted to queue <tb>.
<<Waiting for dispatch ...>>
<<Starting on gb-0004>>
[0] Intel(R) Trace Collector INFO: Writing tracefile a.out.stf in /home/b/b59999/
```

3. Intel Trace Analyzer / Collector でのトレース結果確認
   Exceed onDemand など X（GUI）が利用できる環境で、出力された **stf**ファイル を指定して **traceanalyzer** コマンドを実行するとIntel Trace Analyzerが起動します。

```
$ traceanalyzer a.out.stf
```

マニュアル

Ver.2018

- Intel Trace Analyzer and Collector for Linux

Ver.2019

- Intel Trace Analyzer for Linux Reference Guide
- Intel Trace Collector for Linux Reference Guide

利用可能なソフトウェア

数値計算ライブラリ一覧

- Intel MKLライブラリ
- NAGライブラリ
- IMSLライブラリ

ISVアプリケーション一覧

ソフトウェアベンダーが開発しているアプリケーションの一覧です。 GUIアプリケーションを利用する際には、X11環境が必要になります。

- AVS/Express
- ENVI/IDL
- Tecplot360
- Gaussian16 / GaussView6
- Maple
- Mathematica
- MATLAB
- MSC Adams
- MSC Nastran
- MSC Patran
- MSC Marc
- MSC Marc Mentat
- LS-DYNA
- ANSYS

利用可能なソフトウェア一覧

システムごとに利用可能なソフトウェアの一覧とモジュールファイル名を掲載しています。 最新情報の反映が遅れる場合がありますのでご留意ください。 ログインノード上で module avail コマンドを実行することで、一覧を表示することも可能です。

Module File が none となっているソフトウェアは、追加の設定なしにご利用いただけます。 それ以外については、module load コマンドで環境設定が必要です。

- + ： 利用可能(ただし各ソフトウェアのライセンスを順守してください)
- AU ： 学術研究機関限定で利用可能
- KU ： 京都大学構成員限定で利用可能
- - ： 利用不可

Search for names..

Table Importer: Could not resolve file name 'software.csv type=csv id=myTable raw="true"]'.

Intel MKLライブラリ

MKLライブラリとは

MKLライブラリは、工学、科学、金融系ソフトウェアの開発者向けに、線形代数ルーチン、高速フーリエ変換、ベクトル・マス・ライブラリー関数、乱数生成関数を提供します。これらのルーチンや関数はすべて、Intelプロセッサ用に最適化されています。また、その他のx86プロセッサにも対応しており、問題なく動作します。

利用環境

利用できるバージョン・システム

バージョン	モジュールファイル名	システムA	システムB/C	システムG	クラウドシステム	備考
2024.0	intel/2024.0	+	+	-	+	2024年4月導入
2023.2 (default)	intel/2023.2	+	+	-	+	2024年4月導入
2023.1	intel/2023.1	+	+	-	+	2023年8月導入
2022.3	intel/2022.3	+	+	-	+	2022年11月導入

+ ： 利用可能
- ： 利用不可

Intelコンパイラのモジュールファイルをロードするだけで**MKL**もロードされます。 MKLのために特にモジュールをロードする必要はありません。

MKLライブラリを利用する場合は、Intelコンパイラを利用する必要があります。環境設定については、 Intelコンパイラ をご覧ください。

moduleコマンドの詳細は Modules をご覧ください。

利用方法

コンパイル時に、 -qmkl オプションを付与します。

```
$ icc sample.c -qmkl          # C言語の例
$ icpc sample.cpp -qmkl    # C++の例
$ ifort sample.f90 -qmkl    # Fortranの例
```

マニュアル

- Developer Reference for Intel oneAPI Math Kernel Library - C
- Developer Reference for Intel oneAPI Math Kernel Library - Fortran
- Developer Guide for Intel oneAPI Math Kernel Library for Linux*

アドバイザー

- Link Line Advisor

リンク

- Intel oneAPI Math Kernel Library

NAGライブラリ

NAGライブラリとは

NAGライブラリは、1000種類以上の機能をもち、非常に信頼性が高い科学技術計算および統計計算のライブラリです。SMP向けに並列化されたライブラリである NAG Fortran SMP Library (SMP版)とともに、分散メモリ環境で並列科学技術計算を行うための NAG Parallel Library(MPI版)が利用できます。

利用環境

利用できるバージョン・システム

バージョン	モジュールファイル名	システムA	システムB/C	システムG/クラウド
SMP版 Mark 26 (default)	nag_fortran/26	-	AU	-
MPI版 Release 3 (default)	nag_paralell/3	-	AU	-

AU ： 学術研究機関限定で利用可能
- ： 利用不可

環境設定

NAGライブラリを利用する場合は、Intelコンパイラを利用する必要があります。環境設定については、 Intelコンパイラ をご覧ください。

以下のようにmoduleコマンドを実行し、利用したいバージョンのmoduleファイルをロードします。

```
## SMP版を利用する場合
$ module load nag_fortran
--
## MPI版を利用する場合
$ module load nag_parallel
```

SMP版とMPI版の環境を同時に設定することはできません。

moduleコマンドの詳細は Modules をご覧ください。

利用方法

環境設定により、コンパイルに必要なコマンドおよびオプションが環境変数に設定されます。この環境変数を使い、翻訳時にNAGライブラリをリンクしてご利用下さい。

種類	環境変数	備考

種類	環境変数	備考
コンパイルオプション	$NAGFLAGS	環境変数にコンパイルに必要なコンパイルオプションが含まれています。
リンクオプション	$NAGLINK	環境変数にリンクが必要なライブラリが含まれています。

- コンパイル＆リンク例

```
$ ifort $NAGFLAGS  sample.f90  $NAGLINK
```

- 実行例

```
$ tssrun ./a.out
```

サンプルプログラム

NAGライブラリのほぼすべてのルーチンに対して、サンプルのプログラムが用意されています。表に示すディレクトリ配下のファイルをコピーしてご利用ください。

SMP版

サンプルファイル	ファイルパス
プログラム本体	/opt/system/app/nag_fortran/26/fsl6i26dcl/examples/source
入力データ	/opt/system/app/nag_fortran/26/fsl6i26dcl/examples/data
出力結果例	/opt/system/app/nag_fortran/26/fsl6i26dcl/examples/results

MPI版

サンプルファイル	ファイルパス
プログラム本体	/opt/system/app/nag_parallel/3/fdl6i03dcl/examples/source
入力データ	/opt/system/app/nag_parallel/3/fdl6i03dcl/examples/data
出力結果例	/opt/system/app/nag_parallel/3/fdl6i03dcl/examples/results

マニュアル

SMP

- NAGライブラリマニュアル（MARK 26）

MPI

- NAGライブラリマニュアル（Release 3）

リンク

- 日本NAG

IMSLライブラリ

IMSLライブラリとは

「IMSL Fortran ライブラリ」とは、1000以上の数値計算、特殊関数、統計解析用の関数を含む、科学計算用の Fortranサブルーチンライブラリです。LAPACK, ScaLAPACK 及びSuperLU を取り込んでおり、ユーザが使い易いようにインターフェースが整備されています。

線形システムと行列操作、固有システム解析、高速フーリエ変換 (FFT) などの計算時間が掛かるアルゴリズムは、各種のSMPシステム上で並列処理され、性能向上を図ることができます。

利用環境

利用できるバージョン・システム

バージョン	モジュールファイル名	システムA	システムB/C	システムG/クラウド
2022.1.0	imsl/fnl-2022.1.0	-	AU	-

AU ： 学術研究機関限定で利用可能
- ： 利用不可

環境設定

IMSLライブラリを利用する場合は、Intelコンパイラを利用する必要があります。以下のようにmoduleコマンドを実行します。

```
$ module load imsl/fnl-2022.1.0
```

moduleコマンドの詳細は Modules をご覧ください。

利用方法

環境設定により、コンパイルに必要なリンクオプションが環境変数として設定されます。この環境変数を使い、コンパイルを行ってください。

種類	い環境変数	説明
コンパイラ	$FC	Fortran コンパイラ
コンパイルオプション	$F90FLAGS	Fortran90用コンパイルオプション
リンクオプション	$LINK_FNL	静的ライブラリ（逐次）

種類	い環境変数	説明
〃	$LINK_MPIS	静的ライブラリ（MPI）

- コンパイル＆リンク例

```
$ ifort $F90FLAGS sample.f90 $LINK_FNL       （逐次　システムB,C）
$ mpiifort $F90FLAGS sample.f90 $LINK_MPIS   （MPI　システムB,C）
```

リンク

- IMSL
- IMSL FORTRAN NUMERICAL LIBRARY

Python

利用環境

利用できるバージョン・システム

以下の表に示すpython を利用することができます。 いずれもOSに付属するパッケージとして提供されているため、モジュールファイルによる環境設定は不要です。

バージョン	モジュールファイル名	システム A	システム B/C	システム G	クラウドシステム	備考
2.7	none	+	+	+	+	コマンド名: python2.7, pip2.7
3.6	none	+	+	+	+	コマンド名: python3.6, pip3.6
3.8	none	+	+	+	+	コマンド名: python3.8, pip3.8

+ ： すべてのユーザが利用可能
- ： 利用不可

利用可能なライブラリ

標準でインストールされているライブラリは pip コマンドで確認できます。 パッケージの追加が必要な場合は pipコマンドでご自身のHOMEディレクトリ等に追加インストールが可能です。

```
 ## python3.8 環境のインストール済みパッケージ一覧を表示する場合
$ pip3.8 list
Package                       Version
----------------------------- ----------
absl-py                       1.3.0
asgiref                       3.6.0
astunparse                    1.6.3
...
```

機械学習/深層学習

以下のPythonパッケージは予め特定のpython環境にインストールしてあります。 利用するにはモジュールファイルのロードが必要です。

Python	対象システム	種別	パッケージ	バージョン	ライブラリ	モジュールファイル	補足事項

Python	対象システム	種別	パッケージ	バージョン	ライブラリ	モジュールファイル	補足事項
python3.8	A/B/C	CPU	TensorFlow	2.11.0	MKL 2022.3	tensorflow/2.11.0.py38_intel-2022.3	Intel Optimized AVX512版
python3.8	A/B/C	CPU	MXNet	1.6.0	MKL 2022.3	mxnet/1.6.0.py38_intel-2022.3	MKL対応版
python3.8	A/B/C	CPU	PyTorch	2.0.1	MKL 2022.3	pytorch/2.0.1.py38_intel-2022.3	intel extension for pytorch, Intel oneCCL Bindings for Pytorch
python3.8	A/B/C	CPU	PyTorch	1.13.1	MKL 2022.3	pytorch/1.13.1.py38_intel-2022.3	intel extension for pytorch, Intel oneCCL Bindings for Pytorch
python3.8	G	GPU	TensorFlow	2.11.0	CUDA 11.2	tensorflow/2.11.0.py38_cuda-11.2	GPU版 TensorRT利用可
python3.8	G	GPU	MXNet	1.9.1	CUDA 11.7	mxnet/1.9.1.py38_cuda-11.7	GPU版
python3.8	G	GPU	PyTorch	1.13.1	CUDA 11.7	pytorch/1.13.1.py38_cuda-11.7	GPU版

Intel Extension for PyTorch の参考資料

- Documentation
- Examples

システムA/B/C 向けの pytorch に関する補足

システムA/B/C 向けの pytorch は、Intel Extension for Pytorch は Intel oneCCL Bindings for Pytorch を利用します。 以下にサンプルプログラムと、ジョブスクリプトを例を記載していますので、参考にしてください。 各ソフトウェアの使い方の詳細は公式マニュアルをご参照ください。

サンプルプログラム

```
 #!/usr/bin/python3.8

# Reference source
# https://github.com/intel/torch-ccl#usage

import os
import torch.nn.parallel
import torch.distributed as dist
import oneccl_bindings_for_pytorch
import intel_extension_for_pytorch

os.environ['MASTER_ADDR'] = str(os.environ.get('TORCH_MASTER_ADDR', '127.0.0.1'))
os.environ['MASTER_PORT'] = str(os.environ.get('TORCH_MASTER_PORT', '29500'))
os.environ['RANK'] = str(os.environ.get('PMI_RANK', 0))
os.environ['WORLD_SIZE'] = str(os.environ.get('PMI_SIZE', 1))

backend = 'ccl'
dist.init_process_group(backend)
my_rank = dist.get_rank()
my_size = dist.get_world_size()
print("my rank = %d  my size = %d" % (my_rank, my_size))

x = torch.ones([2, 2])
y = torch.ones([4, 4])
with torch.autograd.profiler.profile(record_shapes=True) as prof:
    for _ in range(10):
        dist.all_reduce(x)
        dist.all_reduce(y)
dist.barrier()
print(prof.key_averages(group_by_input_shape=True).table(sort_by="self_cpu_time_total"))
```

ジョブスクリプト

```
 #!/bin/bash
#============ SBATCH Directives =======
#SBATCH -p gr10001b
#SBATCH -t 1:0:0
#SBATCH --rsc p=4:t=1:c=1
#SBATCH -o %x.%A.out

#============ Shell Script ============

. /usr/share/Modules/init/bash
module load pytorch/2.0.1.py38_intel-2022.3
module list

set -x

export CCL_WORKER_COUNT=${OMP_NUM_THREADS}
export CCL_WORKER_AFFINITY=$(numactl -s |grep physcpubind: | sed -e 's/physcpubind: //'  -e 's/ *$//' -e '
export TORCH_MASTER_ADDR=${SLURMD_NODENAME}-ib0

srun python3.8 -u profiling.py
```

ANACONDA DISTRIBUTION

Pythonのパッケージ群が利用可能なANACONDA DISTRIBUTION をインストールしています。 標準のpythonではなく、ANACONDA DISTRIBUTION がお好みの場合は以下のモジュールファイルをロードすることで利用できます。

DISTRIBUTION	**Python**	モジュールファイル
anaconda3 2022.10	python3.9	anaconda3/2022.10

AVS/Express

利用環境

利用できるバージョン・システム

Modulesソフトウェアパッケージは，アプリケーションの利用に必要な環境設定をmoduleコマンドを実行することで，動的に切り替えて設定することができます．また，異なるバージョンのアプリケーションを切り替えて利用する際に，簡単に環境設定を変更することができます．詳細は Modules をご覧ください．

module avail コマンドにて，利用できるモジュールファイル一覧が確認できます．

バージョン	モジュールファイル名	システムA	システムB/C/G	クラウドシステム
8.5.1 (default)	avs/8.5.1	-	AU	-

AU ： 学術研究機関限定で利用可能
- ： 利用不可

利用可能なライセンス数

同時に利用可能なユーザ数は2ライセンスです。

機能概要

AVS/Expressでは，様々な機能がモジュールと呼ばれる部品として提供されています．モジュール同士をマウス操作で組み合わせるだけで，意のままの可視化を簡単に実現します．ユーザーが独自にモジュールを作成することによって機能を追加することも可能です．また，単なる可視化処理にとどまらず，可視化アプリケーションを作成するための機能も持ち備えています．

機能

- 幅広い分野の可視化をサポート
- 構造解析：流体解析などの数値解析データの可視化
- 医療解析：ＣＴ／ＭＲＩなどの画像処理
- 実験／観測データ：地図情報データなどの可視化
- ニーズに合わせた可視化アプリケーションの構築
- モジュール（可視化機能単位）を対話的に組み合わせて、独自の可視化フローを作成
- 可視化システムの業界標準ＡＶＳの資産を継承

利用分野

- ビジュアリゼーション機能
- 線・面コンター表示、等値面生成、矢印によるベクトル表示
- 流線、パーティクルトレース、任意断面スライス
- 要素シュリンク、ボリュームレンダリング

- ビジュアルプログラミング
- ネットワークエディタ
- オブジェクト指向
- グラフィックス表示機能

利用方法

AVS/Express はGUIアプリケーションであるため、X Windowの環境が必要です。

FastX、NICE DCVを使用してログインするか、Windowsでご利用いただけるX11 Forwarding に対応したSSHクライアントソフト(例えばMobaXterm)およびX11サーバをご利用いただき、GUIアプリケーションが起動可能な方法でシステムにログインしてください。

環境設定(moduleコマンドの実行)

moduleコマンドを実行し，環境設定を行います．（利用したいバージョンのmoduleファイルをロードします）

```
$ module load avs
```

起動方法

GUIをお使いいただける環境で次のコマンドを実行するとAVSが起動します。

計算ノードで起動する場合

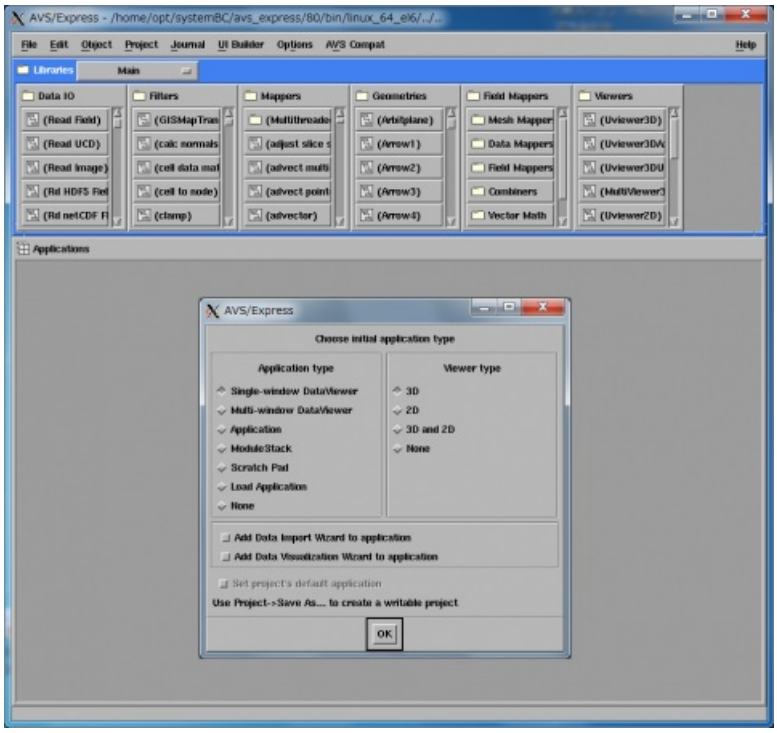
tssrun コマンドで計算ノード上でプロセスを起動し、GUI画面のみログインノードに転送することが可能です。 tssrun コマンドの詳細は 会話型処理 をご覧ください．

```
$ tssrun --x11 xp
```

アプリケーションサーバで起動する場合

GPUを搭載したアプリケーションサーバである app.kudpc.kyoto-u.ac.jp にログインした後に、ノード上で直接起動してください。 サーバ上のGPUを使用したサーバーサイドレンダリングを使用する場合は、NiceDCVをご利用ください。

```
$ xp
```



参考資料

日本語

バージョン 8.5

- デベロッパーズ・ガイド第1部
- デベロッパーズ・ガイド第2部
- モジュール・リファレンス第1部
- モジュール・リファレンス第2部
- チュートリアル・ガイド
- ユーザーズ・ガイド

英語

提供なし

リンク

外部リンク

- AVS 汎用可視化システム（サイバネット）
- AVS （Advanced Visual Systems Inc.）

ENVI/IDL

利用環境

利用できるバージョン・システム

Modulesソフトウェアパッケージは，アプリケーションの利用に必要な環境設定をmoduleコマンドを実行することで，動的に切り替えて設定することができます．また，異なるバージョンのアプリケーション を切り替えて利用する際に，簡単に環境設定を変更することができます．詳細は Modules をご覧ください．

module avail コマンドにて，利用できるモジュールファイル一覧が確認できます．

ENVI

バージョン	モジュールファイル名	システムA	システムB/C/G	クラウドシステム
ENVI 6.0.0 (default)	envi/6.0.0	-	AU	-
ENVI 5.6.3	envi/5.6.3	-	AU	-

IDL

バージョン	モジュールファイル名	システムA	システムB/C/G	クラウドシステム
IDL 9.0.0 (default)	idl/9.0.0	-	AU	-
IDL 8.8.3	idl/8.8.3	-	AU	-

AU ： 学術研究機関限定で利用可能
- ： 利用不可

サポートの終了した古いバージョンの公開は年度末の保守のタイミングで提供終了することがあります。

利用可能なライセンス数

同時に利用可能なユーザ数はENVI/IDL合わせて約10ライセンスです。

ライセンスサービス

ENVI/IDL を手持ちのパソコンにインストールして利用できるライセンスサービスの提供を行っています。
詳細は、 ENVI/IDL（ライセンスサービス） をご覧ください。

利用登録

ENVI/IDLをご利用いただくには、 利用者ポータル から利用登録が必要です。
なお利用登録の際、関係法令等を理解した上で同意していただく必要があります。

ENVI/IDLの利用について

ENVI/IDL につきまして、販売元のHarris Geospatial株式会社より、
本ソフトウエアはU.S. Export Administration Regulations(EAR)を含む米国の輸出管理に関する法令の規制対象であ

り、End User License Agreement および Export Classification の内容に合意した利用者のみ利用可能にするよう依頼がありました。

利用者の皆様におかれましては、関係法令等を理解した上でご利用いただきますよう、お願い致します。

本ソフトウェアは、米国が輸出制限措置を講じている国もしくはその国民・居住者においてダウンロードし、またはこれらの国もしくはその国民・居住者に向けて輸出もしくは再輸出できません。
米国輸出管理法では、日本国内においても米国原産の技術情報やソースコードを日本以外の国籍者(永住権を持つ者を除く）に開示する場合には、その外国籍者の本国への再輸出(「みなし再輸出」）とみなされますので、ご注意ください。

機能概要

機能

ENVI

ENVIは衛星画像、航空写真、レーダ、ハイパースペクトルなどのリモートセンシングデータの解析および可視化を行う統合アプリケーションソフトです。様々な汎用的センサー・データの読み込み、フィルタリング等の画像処理、データ分類、標高データ解析などの機能が利用できます。また、IDLを利用する事で、ENVIにユーザー独自のアルゴリズムを追加することが可能です。

ENVI5.0では、メニューやオプションを一つのインターフェース内に統合されたため、GUIに大きな変更があります。
ENVI4.8以前と同じGUI形式は、ENVI-Classicとして利用することが出来ます。

IDL

IDLは計測・実験、数値計算、統計解析、シミュレーション、ビジュアライゼーションなどで使用されているソフトウェアで、データ解析、可視化、アプリケーション開発に利用されています。様々なファイル形式のデータを読み込む事ができ、600種類以上の各種処理関数／ライブラリが利用できます。また、IDL言語でアプリケーションを作成し、コードをほかのユーザーに配布する事が可能です。

オプション

ENVIオプションモジュール

ご利用いただけるオプションモジュールはありません。

利用分野

ENVI

気象・海洋分野、環境・森林分野、農業分野、地形・地質学、自然資源探索、航空宇宙分野　など

IDL

リモートセンシング、宇宙科学、医用画像、気象　など

利用方法

前提条件

ENVI および IDL はGUIアプリケーションであるため、X Windowの環境が必要です。

FastX、NICE DCVを使用してログインするか、Windowsでご利用いただけるX11 Forwarding に対応したSSHクライアントソフト(例えばMobaXterm)お よびX11サーバをご利用いただき、GUIアプリケーションが起動可能な方法でシステムにログインしてください。

ENVI

環境設定(moduleコマンドの実行)

moduleコマンドを実行し、環境設定を行います。（利用したいバージョンのmoduleファイルをロードします）

```
$ module load envi
```

計算ノードで起動する場合

tssrun コマンドで計算ノード上でプロセスを起動し、GUI画面のみログインノードに転送することが可能です。 tssrun コマンドの詳細は 会話型処理 をご覧ください.

```
$ tssrun --x11 envi
```

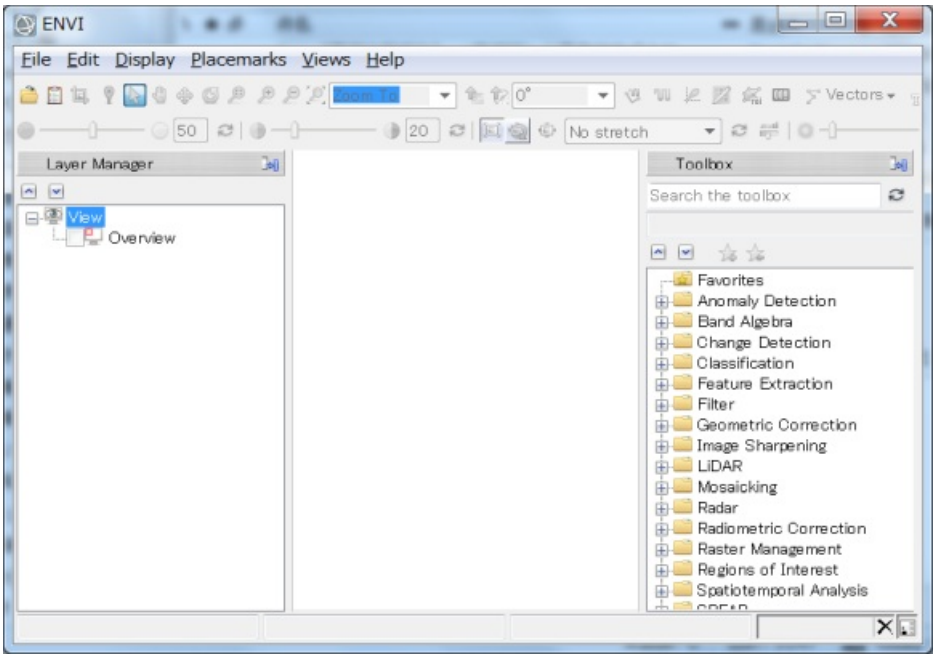
アプリケーションサーバで起動する場合

GPUを搭載したアプリケーションサーバである app.kudpc.kyoto-u.ac.jp にログインした後に、ノード上で直接起動してください。 サーバ上のGPUを使用したサーバーサイドレンダリングを使用する場合は、NiceDCVをご利用ください。

```
$ envi
```

起動画面及び終了方法

起動に成功すると次のような画面が表示されます。



ENVI内のコマンドウィンドウで次のコマンドを実行するとENVIが終了します。

```
$ exit
```

または、「メニューバー」の「File」内にある「Exit」をクリックし、「Terminate this ENVI Session?」で「Yes」を選択するとENVIが終了します。

なお、ENVI Classic UIは下記のコマンドで起動できます。

```
$ tssrun --x11 envi -classic
```

IDL

環境設定（moduleコマンドの実行）

moduleコマンドを実行し、環境設定を行います。（利用したいバージョンのmoduleファイルをロードします）

```
$ module load idl
```

IDL（GUI）

計算ノードで起動する場合

tssrunコマンドで計算ノード上でプロセスを起動し、GUI画面のみログインノードに転送することが可能です。 **tssrun** コマンドの詳細は 会話型処理 をご覧ください.

```
$ tssrun -x11 idlde
```

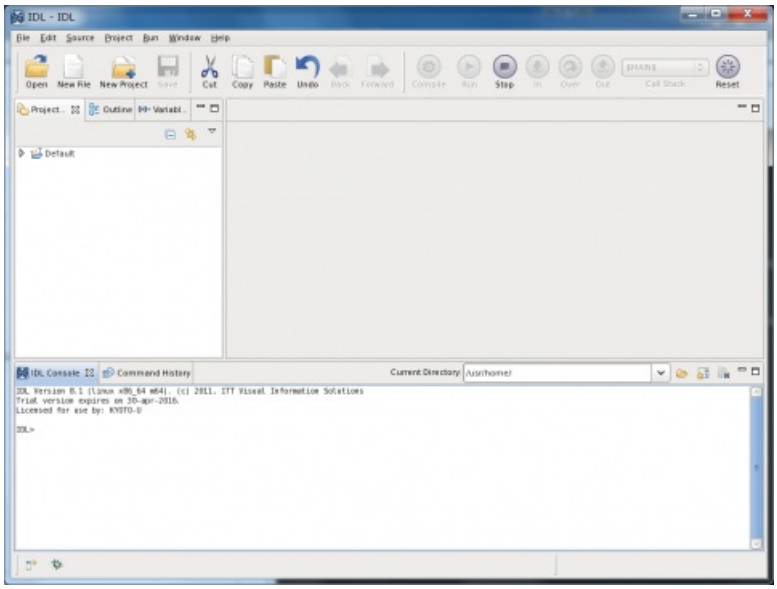
アプリケーションサーバで起動する場合

GPUを搭載したアプリケーションサーバである app.kudpc.kyoto-u.ac.jp にログインした後に、ノード上で直接起動してください。 サーバ上のGPUを使用したサーバーサイドレンダリングを使用する場合は、NiceDCVをご利用ください。

```
$ idlde
```

起動画面及び終了方法

起動に成功すると次のような画面が表示されます。



「メニューバー」の「File」内にある「Exit」をクリックしするとIDLが終了します。

IDL(CUI)

次のコマンドを実行するとCUI版のIDLを起動することができます。 **tssrun** コマンドの詳細は 会話型処理 をご覧ください。

```
$ tssrun idl
```

次のコマンドを実行するとIDLが終了します。

```
> exit
```

IDLオンラインヘルプ

IDLのコンソールで、次のコマンドを実行するとIDLオンラインヘルプが起動します。

```
> idlhelp
```

メニューバーの「File」内にある「Quit」をクリックしするとIDLオンラインヘルプが終了します。

リンク

外部リンク

- Harris Geospatial Solutions
- Harris Geospatial Solutions: End User License Agreement
- Harris Geospatial Solutions: Export Classification

Tecplot360

利用環境

利用できるバージョン・システム

Modulesソフトウェアパッケージは，アプリケーションの利用に必要な環境設定をmoduleコマンドを実行することで，動的に切り替えて設定することができます．また，異なるバージョンのアプリケーション を切り替えて利用する際に，簡単に環境設定を変更することができます．詳細は Modules をご覧ください．

module avail コマンドにて，利用できるモジュールファイル一覧が確認できます．

バージョン	モジュールファイル名	システムA	システムB/C/G	クラウドシステム
2022 R2 (default)	tecplot360/2022R2	-	KU	-
2022 R1	tecplot360/2022R1	-	KU	-

KU ： 京都大学構成員限定で利用可能
- ： 利用不可

利用可能なライセンス数

同時に利用可能なユーザ数は3ライセンスです。

機能概要

数値シミュレーションとＣＦＤの可視化機能ツールであるTecplot360は、科学や、工学のあらゆるデータを素早く思い通りにプロットやアニメーションに作成することができます。複雑なデータの分析や探索、XY、2D、3D タイプの各種プロットを組み合わせた図表の作成、研究結果の情報交換、高画質出力などの多彩なタスクをこなし、データ分析やプロットの定型作業は、自動化することで手間と時間を節減します。

利用方法

Tecplot360 はGUIアプリケーションであるため、X Windowの環境が必要です。

FastX、NICE DCVを使用してログインするか、Windowsでご利用いただけるX11 Forwarding に対応したSSHクライアントソフト(例えばMobaXterm)およびX11サーバをご利用いただき、GUIアプリケーションが起動可能な方法でシステムにログインしてください。

環境設定(moduleコマンドの実行)

moduleコマンドを実行し、環境設定を行います。（利用したいバージョンのmoduleファイルをロードします）

```
$ module load tecplot360
```

起動方法

計算ノードで起動する場合

tssrun コマンドで計算ノード上でプロセスを起動し、GUI画面のみログインノードに転送することが可能です。 tssrun コマンドの詳細は 会話型処理 をご覧ください。

```
$ tssrun --x11 tec360
```

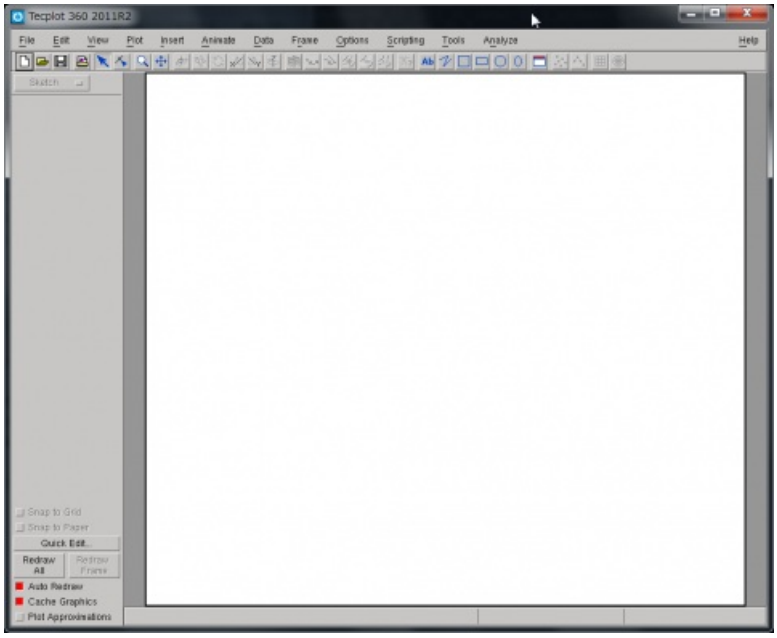
アプリケーションサーバで起動する場合

GPUを搭載したアプリケーションサーバである app.kudpc.kyoto-u.ac.jp にログインした後に、ノード上で直接起動してください。 サーバ上のGPUを使用したサーバーサイドレンダリングを使用する場合は、NiceDCVをご利用ください。

```
$ tec360
```

起動画面及び終了方法

起動に成功すると次のような画面が表示されます。



「メニューバー」の「File」内にある「Exit」を選択するとtecplotが終了します。

Tecplotで大規模データを取り扱う場合

Tecplotは処理中にテンポラリファイルを出力します。デフォルト設定では、/tmp に「tecplot_ユーザ名」という名前のディレクトリを作成し、テンポラリファイルを置くよう設定しています。
数十GB以上の大規模データを取り扱う場合には、デフォルトの/tmpディレクトリでは容量が足りずに異常終了する可能性がありますので、LARGEディスク領域などの十分に余裕のある保存領域を指定してください。

以下のコマンドにて、テンポラリファイルの保存先をLARGEディスク領域に変更できます。

tcshの場合

```
$ setenv TMPDIR /LARGE0/gr19999/b59999
```

bash/zshの場合

```
$ export TMPDIR=/LARGE0/gr19999/b59999
```

Tecplotが正常に動作しない場合

以下の現象が発生した場合、tecplot の起動時に -mesa オプションを加えて実行すると、正常に動作することがあります。

- Tecplotが起動できない（Failed to create OpenGL Contextと表示される）
- TIFFやBMPにエクスポートしようとするとエラーになる

なお、-mesa オプションはOpenGL グラフィック・ライブラリーの機能をソフトでエミュレーションするものです。そのため、描画速度が少し遅くなります。

また、エクスポートした画像の透過表示がおかしくなる場合、下記の手順で設定を変更すると、正常に動作することがあります。

1. 上部メニューの Options => Performance を選択し、設定画面を開く
2. Image Export Options を 「Fast」に変更
3. OKボタンで変更を反映

参考資料

日本語

- Tecplot 日本語ガイド （HULINKS, Inc.）
- Tecplot Tips （HULINKS, Inc.）

英語

- Tecplot 360 Support Product Documentation（Tecplot, Inc.）

リンク

外部リンク

- Tecplot（Tecplot, Inc.）
- Tecplot（HULINKS）

Gaussian16 / GaussView6

利用環境

利用できるバージョン・システム

Modulesソフトウェアパッケージは，アプリケーションの利用に必要な環境設定をmoduleコマンドを実行することで，動的に切り替えて設定することができます．また，異なるバージョンのアプリケーション を切り替えて利用する際に，簡単に環境設定を変更することができます．詳細は Modules をご覧ください．

module avail コマンドにて，利用できるモジュールファイル一覧が確認できます．

Gaussian

並列実行はノード内並列のみサポートします。

バージョン	モジュールファイル名	システムA	システムB/C/G	クラウドシステム
Gaussian16 Rev C.02 (default)	gaussian16/c02	+	+	-

GaussView

バージョン	モジュールファイル名	システムA	システムB/C/G	クラウドシステム
GaussView6 Rev 6.1.1 (default)	gaussian16/c02	+	+	-

+ ： すべてのユーザが利用可能
AU ： 学術研究機関限定で利用可能
KU ： 京都大学構成員限定で利用可能
- ： 利用不可

利用登録

Gaussianをご利用いただくには、利用者ポータル から利用登録が必要です。

機能概要

電子構造モデリング・プログラムであるGaussianは、量子力学の基本法則から、さまざまな条件における分子や反応の特性を研究・予測することが可能です。実験による観測が困難や不可能な特性の研究に利用できます。

機能

Gaussian では、バージョンアップごとに、ONIOM法の機能拡張、溶媒和モデルの機能拡張、新しい汎関数の追加など多くの新機能が追加され、また、並列計算機能の強化などがはかられています。詳しくは、以下のOfficial Gaussian Web Siteの情報を参照ください。

- シングルポイントエネルギー計算
  - 分子系のエネルギー計算、分子軌道の計算、多極子モーメントと原子電荷の予測
- 構造最適化

    - ポテンシャルエネルギー、最適化と収斂条件、最小化、遷移構造の探索
- 振動数計算
    - 振動数の計算、基準振動の解釈、停留点のキャラクタリゼーション
- ONIOMに関する機能
    - 高精度のレイヤーでのモデリングプロセス
    - 分子力学力場のカスタマイズ及び、効率的なONIOM計算
    - 電気特性、電磁特性のONIOM計算

利用分野

- 化学
- 化学工学
- 生物化学
- 物理学など

利用方法(Gaussian)

環境設定(moduleコマンドの実行)

moduleコマンドを実行し、環境設定を行います。（利用したいバージョンのmoduleファイルをロードします）

例

```
$ module load gaussian16
```

実行

バッチジョブの利用

ノード内4並列で実行するバッチジョブスクリプトの例です。バッチの利用方法は バッチ処理 をご覧ください。

```
#!/bin/bash
#============ Slurm Options ===========
#SBATCH -p gr19999b
#SBATCH -t 1:00:00
#SBATCH --rsc p=1:t=4:c=4:m=120G
#============ Shell Script ============
set -x

srun g16 test0000.com
```

入力ファイルでの並列数やメモリの指定

ノード内並列実行する場合は、入力ファイルにも `%NprocShared=` を記述し、並列数を明示する必要があります。また、使用するメモリ量は入力ファイルにも `%Mem=` で明示する必要があります。なお、Gaussianのメモリ使用量は、

%Memを厳密に守らずオーバーすることがあるため、%Memはジョブスクリプトで指定したメモリ量より若干少なめに指定して下さい。

- 例：4並列、メモリ12GB使用したい場合

```
%NprocShared=4
%Mem=12GB
```

subgコマンドでのバッチジョブ実行

subgコマンドは、キュー名、入力ファイルなどを指定し、バッチジョブを投入するコマンドです。ジョブスクリプトを書くよりも簡易にバッチジョブ投入が行えます。

形式

```
$ subg16 queue_name input_file [slurm_command_option ...]
```

例 gr19999b のキューでノード内4並列、メモリ合計30Gバイトを指定

```
$ subg16 gr19999b water.com --rsc p=1:t=4:c=4:m=30G -t 1:00:00
```

計算の途中再開

Gaussianでは、ウォールタイム制限などで途中で終了した計算を途中から続けることができます。

入力データ

次のようにオプションを指定します。「チェックポイントファイル名」には、任意のファイル名を指定してください。

```
%CHK=チェックポイントファイル名.chk
#OPT=RESTART
```

実行

特別な操作は不要です。 入力ファイルに %CHK=チェックポイントファイル名.chk のオプションがある場合、計算が中断されるとchkファイルが生成されます。 次回実行時、入力ファイルに #OPT=RESTART がある場合、chkファイルを読み込んで、前回計算が中断したところから再開されます。

利用方法(GaussView)

GaussViewの起動

```
$ tssrun --x11 gview.sh
```

なお、GaussViewはGUIアプリケーションのため、MobaXterm または FastXを利用してスパコンに接続してください。

GaussViewが正常に動作しない場合

GaussView の起動時に -soft オプションを加えてソフトウェアレンダリングを有効にして実行すると、動作することがあります。

```
$ tssrun --x11 gview.sh -soft
```

参考資料

日本語

提供なし

英語

- Gaussian 16 User Refenrence （Gaussian, Inc.）

リンク

外部リンク

- Official Gaussian Website （Gaussian, Inc）

Gaussian09

利用環境

利用できるバージョン・システム

Modulesソフトウェアパッケージは、アプリケーションの利用に必要な環境設定をmoduleコマンドを実行することで、動的に切り替えて設定することができます。また、異なるバージョンのアプリケーションを切り替えて利用する際に、簡単に環境設定を変更することができます。詳細は Modules をご覧ください。

module avail コマンドにて、利用できるモジュールファイル一覧が確認できます。

GaussView5はシステム非対応のため、利用できません。

Gaussian09

バージョン	モジュールファイル名	システムA	システムB/C/G	クラウドシステム
Gaussian09 Rev E.01(default)	gaussian09/e01	+	+	-

+ ： すべてのユーザが利用可能
AU ： 学術研究機関限定で利用可能
KU ： 京都大学構成員限定で利用可能
- ： 利用不可

利用登録

Gaussianをご利用いただくには、利用者ポータル から利用登録が必要です。

機能概要

電子構造モデリング・プログラムであるGaussianは、量子力学の基本法則から、さまざまな条件における分子や反応の特性を研究・予測することが可能です。実験による観測が困難や不可能な特性の研究に利用できます。

機能

Gaussianでは、バージョンアップごとに、ONIOM法の機能拡張，溶媒和モデルの機能拡張、新しい汎関数の追加など多くの新機能が追加され、また、並列計算機能の強化などがはかられています。詳しくは、以下のOfficial Gaussian Web Siteの情報を参照ください。

- シングルポイントエネルギー計算
  - 分子系のエネルギー計算、分子軌道の計算、多極子モーメントと原子電荷の予測
- 構造最適化
  - ポテンシャルエネルギー、最適化と収斂条件、最小化、遷移構造の探索
- 振動数計算
  - 振動数の計算、基準振動の解釈、停留点のキャラクタリゼーション
- ONIOMに関する機能
  - 高精度のレイヤーでのモデリングプロセス

- 分子力学力場のカスタマイズ及び、効率的なONIOM計算
- 電気特性、電磁特性のONIOM計算

利用分野

- 化学
- 化学工学
- 生物化学
- 物理学など

利用方法(Gaussian 09)

環境設定(moduleコマンドの実行)

moduleコマンドを実行し，環境設定を行います．（利用したいバージョンのmoduleファイルをロードします）

```
$ module load gaussian09
```

実行

バッチジョブの利用

バッチジョブスクリプトの例です。

```
#!/bin/bash
#============ Slurm Options ===========
#SBATCH -p gr10001d     #ジョブキュー(パーティション)の指定
#SBATCH --rsc p=1:t=4:c=4:m=1G
#============ Shell Script ============
set -x

export OMP_PROC_BIND=true

srun g09 test0000.com
```

入力ファイルでの並列数やメモリの指定

ノード内並列実行する場合は，入力ファイルにも `%NprocShared=` を記述し，並列数を明示する必要があります．また，使用するメモリ量は入力ファイルにも `%Mem=` で明示する必要があります．なお，Gaussianのメモリ使用量は，%Memを厳密に守らずオーバーすることがあるため，%Memはジョブスクリプトで指定したメモリ量より若干少なめに指定して下さい．

- 例：4並列，メモリ12GB使用したい場合

```
%NprocShared=4
%Mem=12GB
```

subg09コマンドでのバッチジョブ実行

subg09コマンドは、キュー名、入力ファイルなどを指定し、バッチジョブを投入するコマンドです。ジョブスクリプトを書くよりも簡易にバッチジョブ投入が行えます。

形式

```
subg09 queue_name input_file  [-scrdir dir] [qsub_command_option ...]
```

例 gr19999b のキューでノード内4並列、メモリ合計30Gバイトを指定

```
$ subg09 gr19999b water.com --rsc p=1:t=4:c=4:m=30G
```

リンク

外部リンク

- Official Gaussian Website （Gaussian, Inc）

Maple

利用環境

利用できるバージョン・システム

Modulesソフトウェアパッケージは，アプリケーションの利用に必要な環境設定をmoduleコマンドを実行することで，動的に切り替えて設定することができます．また，異なるバージョンのアプリケーション を切り替えて利用する際に，簡単に環境設定を変更することができます．詳細は Modules をご覧ください．

module avail コマンドにて，利用できるモジュールファイル一覧が確認できます．

バージョン	モジュールファイル名	システムA	システムB/C/G	クラウドシステム
Maple 2024.0 (default)	maple/2024.0	-	KU	-
Maple 2023.0	maple/2023.0	-	KU	-
Maple 2022.0	maple/2022.0	-	KU	-

KU ： 京都大学構成員限定で利用可能
- ： 利用不可

機能概要

Mapleは、多くの数学コマンドとグラフィック機能を備えた数式処理ソフトウェアです。

機能

- 数式処理、代数演算
- ２次元、３次元プロットのグラフ表示
- 高度で汎用性のある数式計算機能をシンプルな操作で利用
- 計算ノウハウを再現性のある技術文章として管理
- 4000種を超える計算機能が標準で用意

利用分野

- ビジュアリゼーション機能
- 線・面コンター表示、等値面生成、矢印によるベクトル表示
- 流線、パーティクルトレース、任意断面スライス
- 要素シュリンク、ボリュームレンダリング
- ビジュアルプログラミング
- ネットワークエディタ
- オブジェクト指向
- グラフィックス表示機能

利用方法

環境設定(moduleコマンドの実行)

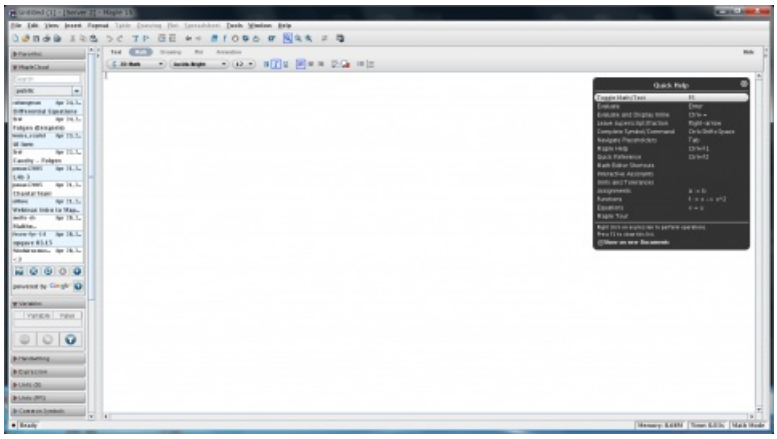
module コマンドを実行し、環境設定を行います。（利用したいバージョンのmoduleファイルをロードします）

```
$ module load maple
```

GUIモード

X Window SystemによるGUIを利用可能な環境で、次のコマンドを実行するとmapleが起動します。 tssrun コマンドの詳細は 会話型処理 をご覧ください。

```
$ tssrun --x11 xmaple
```



処理画面の中央にある画面 ワークシート で会話形式で処理が行えます。処理画面は、 ワークシート 以外に画面の上から メニューバー 、 ツールバー 、 コンテキストバー で構成されています。

終了は、 メニューバー の **File** 内にある **Exit** をクリックします。

CUIモード

次のコマンドを実行するとmapleが起動します。 **tssrun** コマンドの詳細は 会話型処理 をご覧ください。

```
$ tssrun maple
```

次のコマンドを実行するとmapleが終了します。

```
$ quit
```

参考資料

日本語

- コマンドリファレンス（サイバネット）

英語

提供なし

リンク

外部リンク

- Maple（サイバネットシステム）
- Maple Application Center（Maplesoft）

Mathematica

利用環境

利用できるバージョン・システム

Modulesソフトウェアパッケージは，アプリケーションの利用に必要な環境設定をmoduleコマンドを実行することで，動的に切り替えて設定することができます．また，異なるバージョンのアプリケーション を切り替えて利用する際に，簡単に環境設定を変更することができます．詳細は Modules をご覧ください．

module avail コマンドにて，利用できるモジュールファイル一覧が確認できます．

バージョン	モジュールファイル名	システムA	システムB/C/G	クラウドシステム
14.0.0 (default)	mathematica/14.0.0	-	KU	-
13.1.0	mathematica/13.1.0	-	KU	-

KU ： 京都大学構成員限定で利用可能
- ： 利用不可

利用可能なライセンス数

同時に利用可能なユーザ数は3ライセンスです。

機能概要

Mathematicaは、数式処理、数値計算の機能に加え、優れたグラフィック機能を統合的に扱えるシステムです。また、数学の考え方と同じ感覚でプログラミングすることができ、インタラクティブに計算結果が得られます。 Mathematicaを使えばデータ解析や複雑な微分方程式の解を求めるような単一のタスクから、包括的なソリューションやプロトタイプ、アプリケーションの開発まで果たすことができます。

機能

- 数百・数千にもおよぶ高度な記号計算
- 方程式の解、微分方程式、問題の数値的・記号的な最適化
- 数値的なモデリングやシミュレーションが可能
- RAD (迅速なアプリケーション開発) 環境
- インタラクティブな技術レポートや電子又は印刷用の文書の作成
- 数式表現を使った技術文書の作成

利用分野

工学、数学、金融、物理、化学、生物など

利用方法

環境設定(moduleコマンドの実行)

moduleコマンドを実行し、環境設定を行います。（利用したいバージョンのmoduleファイルをロードします）

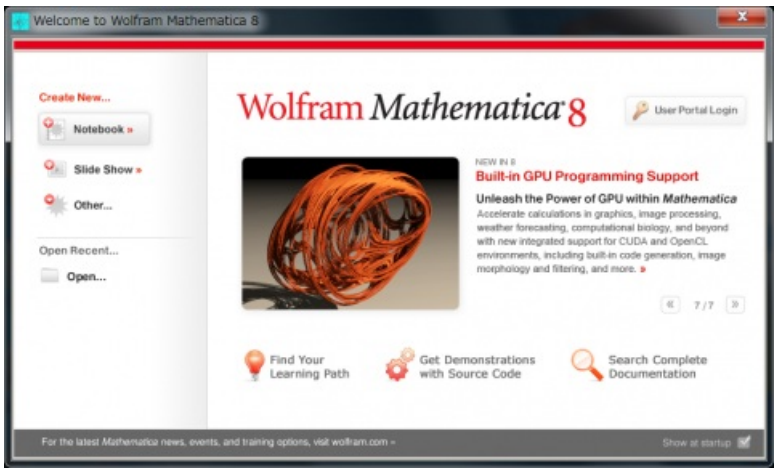
```
$ module load mathematica
```

GUIモード

FastX、NICE DCVを使用してログインするか、Windowsでご利用いただけるX11 Forwarding に対応したSSHクライアントソフト(例えばMobaXterm) およびX11サーバをご利用いただき、GUIアプリケーションが起動可能な方法でシステムにログインしてください。

次のコマンドを実行するとMathematicaが起動します。**tssrun** コマンドの詳細は 会話型処理 をご覧ください。

```
$ tssrun --x11 mathematica
```



起動後、画面左のメニューより、Notebookを選択すると、ノートブック画面が立ち上がります。ノートブック画面でプログラム（数式、計算式）を書き込んで計算させます。

ノートブック画面で、 メニューバー の **File** 内にある **Quit** をクリックするとMathematicaが終了します。

CUIモード

次のコマンドを実行するとMathematicaが起動します。 **tssrun** コマンドの詳細は 会話型処理 をご覧ください。

```
$ tssrun math
```

次のコマンドを実行するとMathematicaが終了します。

```
$ Quit
```

参考資料

日本語

- Documentation Center （Wolfram Research, Inc.）

英語

提供なし

リンク

外部リンク

- Wolfram Mathematica（HULINKS）
- Wolrram Research：Mathematica（WOLFRAM）

MATLAB

利用環境

利用できるバージョン・システム

Modulesソフトウェアパッケージは，アプリケーションの利用に必要な環境設定をmoduleコマンドを実行することで，動的に切り替えて設定することができます．また，異なるバージョンのアプリケーション を切り替えて利用する際に，簡単に環境設定を変更することができます．詳細は Modules をご覧ください．

module avail コマンドにて，利用できるモジュールファイル一覧が確認できます．

バージョン	モジュールファイル名	システムA	システムB/C/G	クラウドシステム
2023a (default)	matlab/R2023a	-	KU	-

KU ： 京都大学構成員限定で利用可能
- ： 利用不可

機能概要

MATLABは、計算、可視化、プログラミング機能を統合した技術計算のための高性能言語です。エンジニアリングや科学技術問題の基本であるベクトルとマトリックスの演算をサポートしています。線形代数をはじめ偏微分方程式など、様々な数学的関数をデータ解析に利用でき、データ可視化のための 2次元／3次元画像関数を備えています。また、C、C++、Fortran といった他の言語のサブルーチンを組み込み関数のように呼び出すことができます。

MATLABでは、基本機能以外に特別なクラスの問題に対応できるように拡張されMATLAB関数を集めたToolboxを用いることにより幅広い分野に対応できるようになっています。

オプション

- **Simulink**

Simulink はマルチドメインシミュレーションやダイナミックシステム、および組込みシステムのモデルベースデザインのためのプラットフォームです。対話型グラフィカル環境およびブロックライブラリ群により、通信、制御、動画像処理、静止画像処理を含め、時間依存システムの設計、シミュレーション、実装、テストが可能です。

利用分野

- 信号・画像処理
- 通信システム設計
- 制御系設計
- 実験計測
- 金融工学におけるモデリングと解析
- コンピュータを利用した生命工学 など

利用可能なツールボックス

MATLAB起動後に、コマンドウィンドウで `ver` コマンドを入力することで確認いただけます。

利用方法

環境設定(moduleコマンドの実行)

moduleコマンドを実行し、環境設定を行います。（利用したいバージョンのmoduleファイルをロードします）

```
$ module load matlab
```

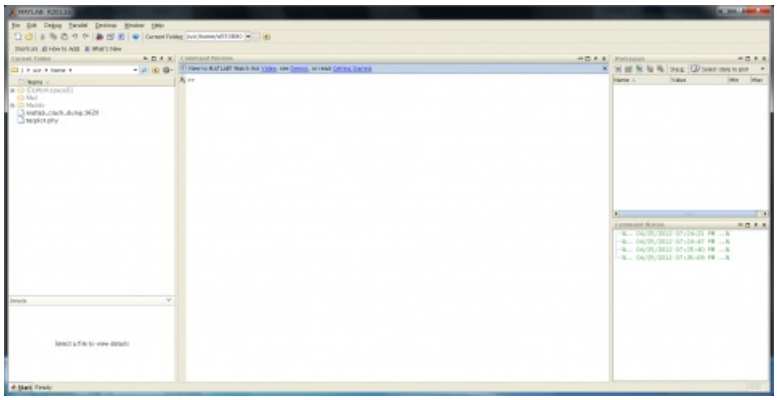
MATLAB

GUIモード

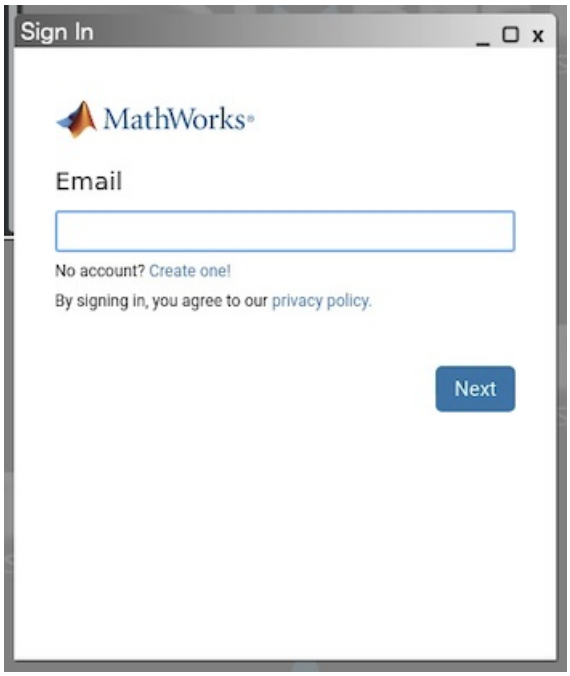
FastX, NiceDCV等X Window Systemが利用可能な環境でログインした後に、次のコマンドを実行するとMATLABが起動します。 tssrun コマンドの詳細は 会話型処理 をご覧ください。

```
$ tssrun --x11 matlab
```

メニューバー の **File** 内にある **Exit MATLAB** をクリックするとMATLABが終了します。



なお、初回起動時は以下のようなログイン画面が表示されます。京都大学構成員の方は全学アドレス (@kyoto-u.ac.jp または @st.kyoto-u.ac.jp)を入力してください。



CUIモード

次のオプションを指定し、コマンドを実行するとMATLABが起動します。 **tssrun** コマンドの詳細は 会話型処理 をご覧ください。

```
$ tssrun matlab -nodesktop -nosplash
```

次のコマンドを実行するとMATLABが終了します。

```
>> quit
```

起動に失敗する場合の対処

MATLABを起動した際に、 **Unable to communicate with required MathWorks services (error 5201).**といったエラーが表示された場合は、以下の手順を行うことで改善されることが ありますので、お試しください。

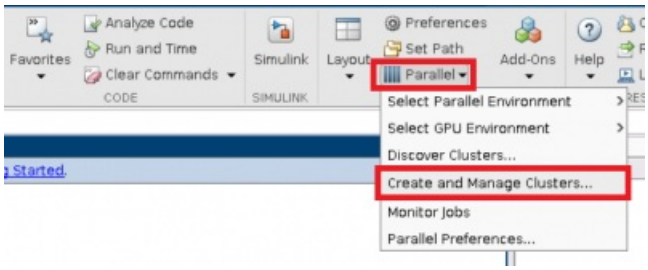
```
$ rm -rf ~/.MathWorks/ServiceHost/*/v[0-9]*
```

MATLAB Parallel Computing Toolbox

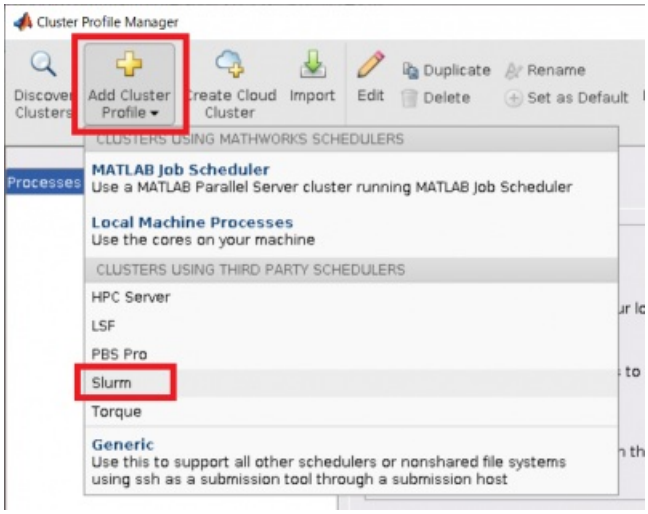
Parallel Computing Toolboxを用いて複数ワーカーで並列処理を行うことで、計算時間を短縮することが可能です。

クラスタープロファイルの設定方法

1. Parallel > Create and Manage Clusters をクリックします。



2. Add Cluster Profile > Slurm をクリックします。



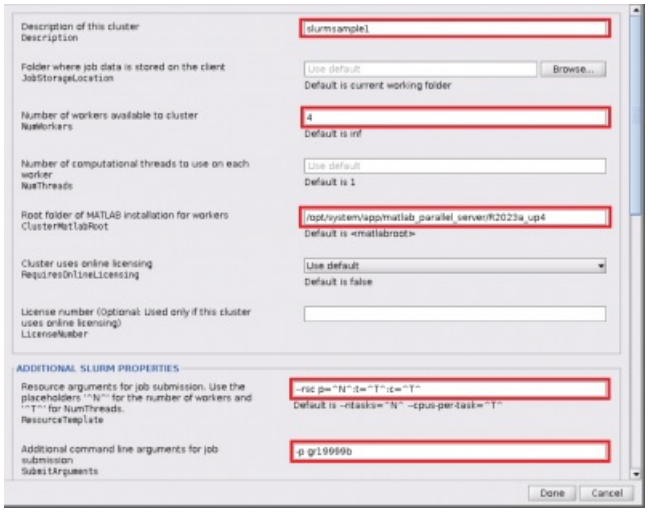
Product Required の画面が出てきた場合、"OK" をクリックしてください。

3. プロファイルの設定画面が出てきますので、次の項目を記入します。
    - **Root folder of MATLAB installation for workers ClusterMatlabRoot**:
      /opt/system/app/matlab_parallel_server/R2023a_up4 （MATLAB 2023aの場合）
    - **Resource arguments for job submission**：
      --rsc p=^N^:t=^T^:c=^T^
    - **Additional command line arguments for job submission Submitarguments**：
      -p <キュー名>

その他の項目は任意です。

Root folder of MATLAB installation for workers ClusterMatlabRootで指定するフォルダは、MATLABのバージョンによって異なります。 下記を参考に指定してください。

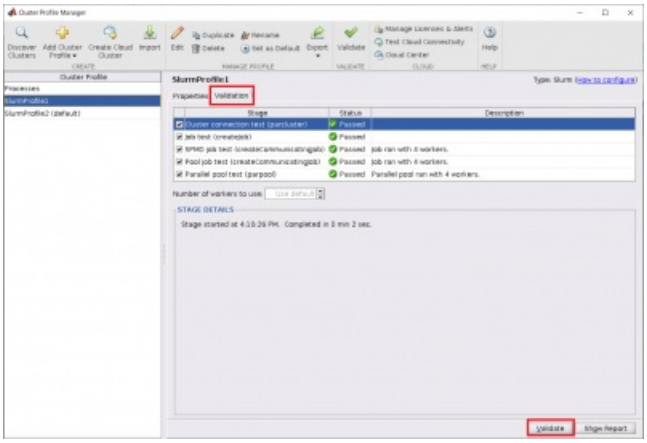
バージョン	フォルダ
2023a	/opt/system/app/matlab_parallel_server/R2023a_up4



4. 設定が終了したら、"Done" をクリックします。

動作確認

1. Validation タブを開いて、"Validate"をクリックします。



2. テストが終了し、"Status"が"Passed"になっていれば問題ありません。"Failed"の場合、"Properties"から設定を見直してください。
3. 設定が終了したら、右上の "×" をクリックしてCluster Profile Manager の画面を閉じます。

クラスタープロファイルの使用方法

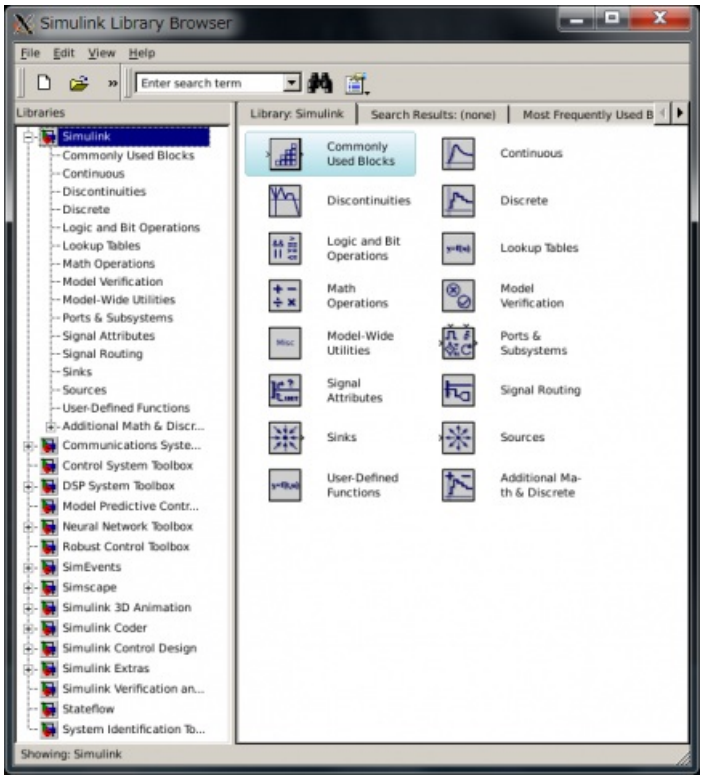
入力ファイル（xxxxxxxx.m）内で、クラスタープロファイル（例：SlurmProfile1）を指定します。

```
c = parcluster('SlurmProfile1')
parpool(c,4)
parfor i = 1:n
    a(i) = max(abs(eig(rand(A))));
end

delete(gcp('nocreate'))
```

Simulink

SimulinkはMATLAB環境と統合されており、MATLABの Command Window から以下のコマンドを入力することで、起動できます。

```
>> simulink
```



次のコマンドを実行するとSimulinkが終了します。

「メニューバー」の「File」内にある「Close」をクリックするとSimulinkが終了します。

MATLAB Compiler

MATLABのmファイルをC/C++のソースファイルに変換してコンパイルし、スタンドアローンで実行可能なファイルを作成することができます。

コンパイル方法

MATLABを起動し、 Command Window から以下のコマンドを入力することで、コンパイルできます。

```
>> mcc -m 入力ファイル名
```

入出力ファイル

入力ファイル名は、xxxxxxxx.m のように、サフィックスに m を付けます。
出力ファイルは以下のようなものが生成されます。

```
「xxxxxxxx 」…実行ファイル
「run_xxxxxxxx.sh」…実行ファイルを起動するためのスクリプト
「xxxxxxxx_mcc_component_data.c」
「xxxxxxxx.prj」
「xxxxxxxx_main.c」
「mccExcludedFiles.log」
「readme.txt」
```

MATLABからの外部プログラムの呼び出し

MEX-ファイルを作成することで、MATLABからCまたはFortranのサブルーチンを組み込み関数のように呼び出すことができます。

MATLABを起動し、 Command Window から以下のコマンドを入力することで、コンパイルできます。

```
>> mex ファイル名
```

参考資料

日本語

- 製品ドキュメンテーション（The MathWorks, Inc.）

英語

- Documentation Center （MathWorks, Inc.）

リンク

外部リンク

- マスワークス公式日本語サイト（The MathWorks, Inc.）

MSC Adams

利用環境

利用できるバージョン・システム

Modulesソフトウェアパッケージは、アプリケーションの利用に必要な環境設定をmoduleコマンドを実行することで、動的に切り替えて設定することができます。また、異なるバージョンのアプリケーションを切り替えて利用する際に、簡単に環境設定を変更することができます。詳細は Modules をご覧ください。

module avail コマンドにて、利用できるモジュールファイル一覧が確認できます。

バージョン	モジュールファイル名	システムA	システムB/C/G	クラウドシステム
2023.4 (default)	adams/2023.4	-	AU	-
2022.3.1	adams/2022.3.1	-	AU	-

+ ： すべてのユーザが利用可能
AU ： 学術研究機関限定で利用可能
- ： 利用不可

利用可能なライセンス数

同時に利用可能な並列数/ユーザ数には上限があります。 ライセンスの不足によるエラーが生じた場合は、ライセンスの利用に関してご協力を依頼することがあります。

機能概要

Adamsは、複数のボディ要素（部品）と拘束要素（ジョイント）で構成されたMBS（Multi Body System）に初期条件を定義することで、各部品に発生する力や運動を時系列にてシミュレートすることができる機構解析ソフトウェアです。なお、Adams 2012 以降では、MD Adams との製品統合が行われ、MD Adamsの機能が使えるようになりました。

機能

Adamsは、汎用的に使用できる基本プロダクトと解析ニーズに対応したプラグインにより構成されています。

- 機構解析用基本アプリケーション
  - Adams Solver（機構モデル解析用ソルバー）
  - Adams View（機構モデル作成用GUI）
  - Adams Exchange（CADデータ読込機能（中間形式））
- 複合領域解析向けプラグイン
  - Adams Control（制御系モデルとのCo-Simulation 機能）
  - Adams Flex（弾性体作成インターフェース（MSC Nastran Master DB 入力も可能））
  - Adams Linear（固有値計算ソルバー（MSC Nastran BDF 出力も可能））
  - Adams Vibration（機構モデルの周波数応答計算ソルバー）
  - Adams Durability（応力リカバリ機能+MSC Fatigue用インターフェース）

- 自動車設計向けプラグイン
  - Adams Car（車両走行解析機能）
  - Adams Tire FTire（乗心地解析用タイヤ）
  - Adams Leafspring Toolkit（リーフスプリング設計専用ツール）
- 機械設計向けプラグイン
  - Adams Gear Advanced Technology Toolkit（弾性体ギア設計専用ツール）
  - Adams Bearing Advanced Technology Toolkit（弾性体ベアリング設計専用ツール）

利用方法

前提条件

Adams はGUIアプリケーションであるため、X Windowの環境が必要です。

FastX、NICE DCVを使用してログインするか、Windowsでご利用いただけるX11 Forwarding に対応したSSHクライアントソフト(例えばMobaXterm)およびX11サーバをご利用いただき、GUIアプリケーションが起動可能な方法でシステムにログインしてください。

環境設定(moduleコマンドの実行)

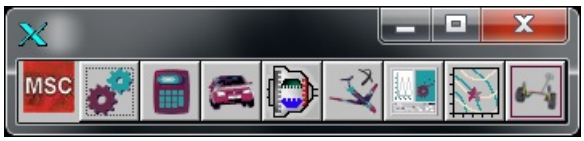
moduleコマンドを実行し、環境設定を行います。（利用したいバージョンのmoduleファイルをロードします）

```
$ module load adams
```

起動方法

次のコマンドを実行するとAdamsが起動します。 tssrun コマンドの詳細は 会話型処理 をご覧ください。

```
$ tssrun --x11 mdi
```



スレッド並列計算の利用

※パーソナルコースキューやグループコースキューをお持ちの方は、並列計算をご利用の際は前述のキューでアプリをご利用ください。

tssrun コマンドを実行の際に、 --rsc オプションでスレッド並列数を指定します。

例1：4並列実行... t,cの値を4に指定します

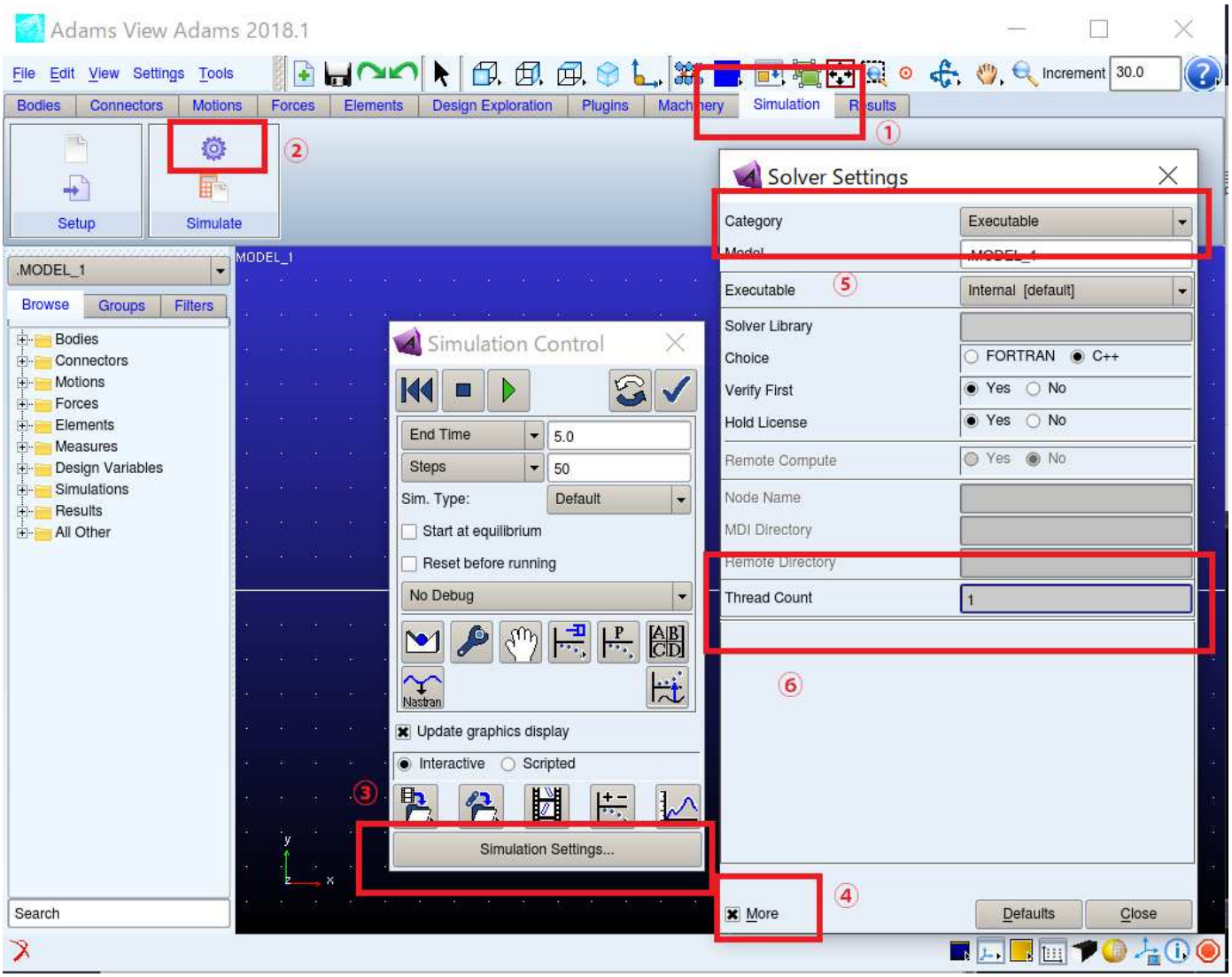
```
$ tssrun --x11 --rsc t=4:c=4 mdi
```

例2：利用可能メモリも増量する場合... mの値も指定します（何も指定しないと3413MBとなります）

```
$ tssrun --x11 --rsc t=4:c=4:m=10G mdi
```

AdamsツールバーからAdams Viewを起動後、解析の実行前に下記の設定を行って、並列数を指定してください。

[Simulation]→[ギアマーク（Simulation Control）]→[Simulation Settings]→[More]→[Category] を "Executable" に
→[Thread Count] を設定（4並列で実行したければ4を入力）

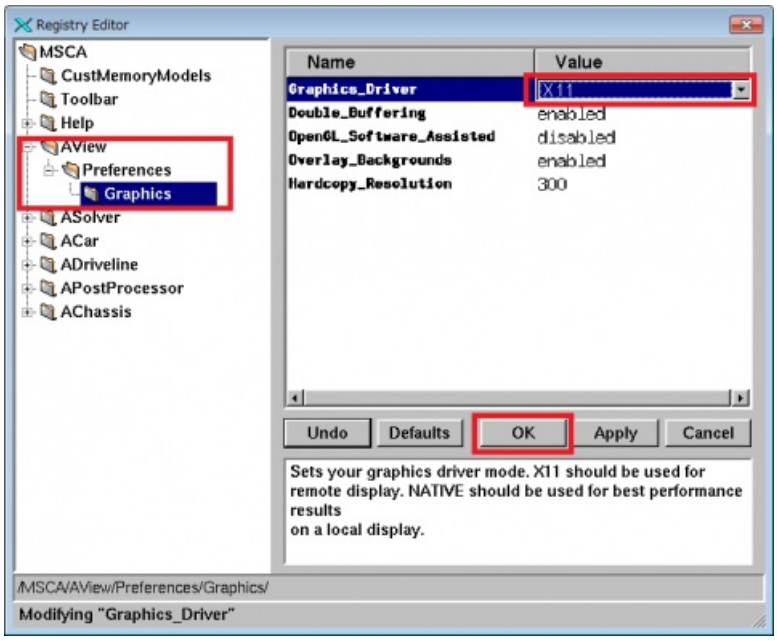


グラフィックスライブラリの設定

以下の手順によりグラフィックスライブラリの設定をX11に切り替えてください。

1. AdamsツールバーのMSCボタンを右クリックし、 **Start Registry Editor** を選択します。
2. **Registry Editor** ダイアログが表示されますので、左側のツリーで、 **AView** => **Preferences** => **Graphics** を選択します。

3. 右側の設定画面で、 **Graphics_Driver** を **X11** に設定し、 **OK** ボタンを押します。



メニューの日本語化

以下の環境変数を設定した上で、Adamsを起動すると、 **Adams/View** のメニューを日本語化することができます。

tcshの場合

```
$ setenv ADAMS_GUI_LOCALE japanese
$ setenv MDI_GUI_FONT_FAMILY -dt-gothic-bold-i-normal
```

bashの場合

```
$ export ADAMS_GUI_LOCALE=japanese
$ export MDI_GUI_FONT_FAMILY=-dt-gothic-bold-i-normal
```

なお、日本語表示に成功した場合でも、Adams/View の Message Window にフォントに関する WARNING が表示されます。

参考資料

日本語

Adams ドキュメント （MSC Software）

英語

Adams Docs （MSC Software）

リンク

外部リンク

Adams （MSC Software）

MSC Nastran

利用環境

利用できるバージョン・システム

Modulesソフトウェアパッケージは、アプリケーションの利用に必要な環境設定をmoduleコマンドを実行することで、動的に切り替えて設定することができます。また、異なるバージョンのアプリケーションを切り替えて利用する際に、簡単に環境設定を変更することができます。詳細は Modules をご覧ください。

module avail コマンドにて、利用できるモジュールファイル一覧が確認できます。

バージョン	モジュールファイル名	システムA	システムB/C/G	クラウドシステム
2023.4 (default)	nastran/2023.4	-	AU	-
2022.3	nastran/2022.3	-	AU	-

+ ： すべてのユーザが利用可能
AU ： 学術研究機関限定で利用可能
- ： 利用不可

利用可能なライセンス数

同時に利用可能な並列数/ユーザ数には上限があります。 ライセンスの不足によるエラーが生じた場合は、ライセンスの利用に関してご協力を依頼することがあります。

機能概要

MSC Nastranは、部品レベルから複雑な構造物全体まで、幅広い解析を行うことができる汎用構造解析プログラムです。基本解析機能に加えて、複合領域シミュレーション機能により、従来の線形静解析、様々な動解析、設計最適化解析、熱解析、3D接触を含む高度な陰解法非線形解析、衝突・衝撃解析、さらに落下解析などの高速現象のための陽解法非線形解析などの領域のシミュレーションを行えます。

機能

- 線形・非線形、静的・動的解析
- 実固有値解析
- 周波数応答解析
- 過渡応答解析
- ランダム応答解析
- 音響-構造連成解析
- 衝撃・衝突解析
- 最適化解析
- Marcとの連成
- Adamsとの連成 など

利用方法

環境設定(moduleコマンドの実行)

moduleコマンドを実行し、環境設定を行います。（利用したいバージョンのmoduleファイルをロードします）

```
$ module load nastran
```

会話型での実行

形式

次のコマンドでプログラムを起動することが可能です。 tssrun コマンドの詳細は 会話型処理 をご覧ください。

```
$ tssrun nast 入力ファイル [オプション ...] batch=no
```

subnastranコマンドでのバッチジョブ実行

ジョブ投入を簡略化するために、subnastranコマンドを用意しています。ジョブスクリプトを記述せずに、バッチ実行を行えます。

形式

```
$ subnastran キュー名 入力ファイル [qsub option ...] [Nastran option ...]
     Nastran option
       smp=(並列数)    Shared Memory Parallel (default:  smp=1)
```

例

```
$ subnastran eb d0307.dat smp=4
```

ジョブの実行が終了後は、入力ファイル以外に新たに以下のような実行結果ファイルが生成されます。

- ジョブ処理結果ファイル
  - B072615.o443917 ---- 標準出力ファイル
  - B072615.e443917 ---- エラーファイル
- 出力ファイル（入力ファイル名と同じ名前で作成されます）
  - d0307.f04 ----実行要約ファイル
  - d0307.log ----ログファイル
  - d0307.f06 ----解析結果ファイル

（注意）同じ入力ファイル名で、ジョブを何度も実行すれば、実行結果ファイル毎に以下のように番号が付与されたファイルがいくつも作成されるのでご注意ください。ジョブを続けて4件実行した場合のファイル名との関係を以下に示します。

- 結果ファイル
  - d0307.f06 （4件目実行ジョブ。最新ジョブ）
  - d0307.f06.1（1件目実行ジョブ。最も古いジョブ）
  - d0307.f06.2（2件目実行ジョブ）
  - d0307.f06.3（3件目実行ジョブ）

参考資料

日本語

- MSC Nastran ドキュメント （MSC Software）

英語

- MSC Nastran Docs （MSC Software）

リンク

外部リンク

- MSC.Nastran （MSC Software）

MSC Patran

利用環境

利用できるバージョン・システム

Modulesソフトウェアパッケージは、アプリケーションの利用に必要な環境設定をmoduleコマンドを実行することで、動的に切り替えて設定することができます。また、異なるバージョンのアプリケーションを切り替えて利用する際に、簡単に環境設定を変更することができます。詳細は Modules をご覧ください。

module avail コマンドにて，利用できるモジュールファイル一覧が確認できます．

バージョン	モジュールファイル名	システムA	システムB/C/G	クラウドシステム
2023.4 (default)	patran/2023.4	-	AU	-
2022.3	patran/2022.3	-	AU	-

+ ： すべてのユーザが利用可能
AU ： 学術研究機関限定で利用可能
- ： 利用不可

利用可能なライセンス数

同時に利用可能な並列数/ユーザ数には上限があります。 ライセンスの不足によるエラーが生じた場合は、ライセンスの利用に関してご協力を依頼することがあります。

機能概要

Patranはシミュレーションを行うエンジニアに使いやすく、豊富な機能を提供するCAE統合環境プリ・ポストソフトウェアです。主要な3次元CADシステムに対応するダイレクトインターフェイスを介して、正確なCAD形状のインポートを行います。さらに、高水準のメッシュ作成機能や可視化機能、柔軟なカスタマイゼーション機能などが利用できます。

利用分野

- 航空宇宙
- 自動車
- 造船
- 医療

利用方法

Patran はGUIアプリケーションであるため、X Windowの環境が必要です。

FastX、NICE DCVを使用してログインするか、Windowsでご利用いただけるX11 Forwarding に対応したSSHクライアントソフト(例えばMobaXterm)およびX11サーバをご利用いただき、GUIアプリケーションが起動可能な方法でシステムにログインしてください。

環境設定(moduleコマンドの実行)

moduleコマンドを実行し、環境設定を行います。（利用したいバージョンのmoduleファイルをロードします）

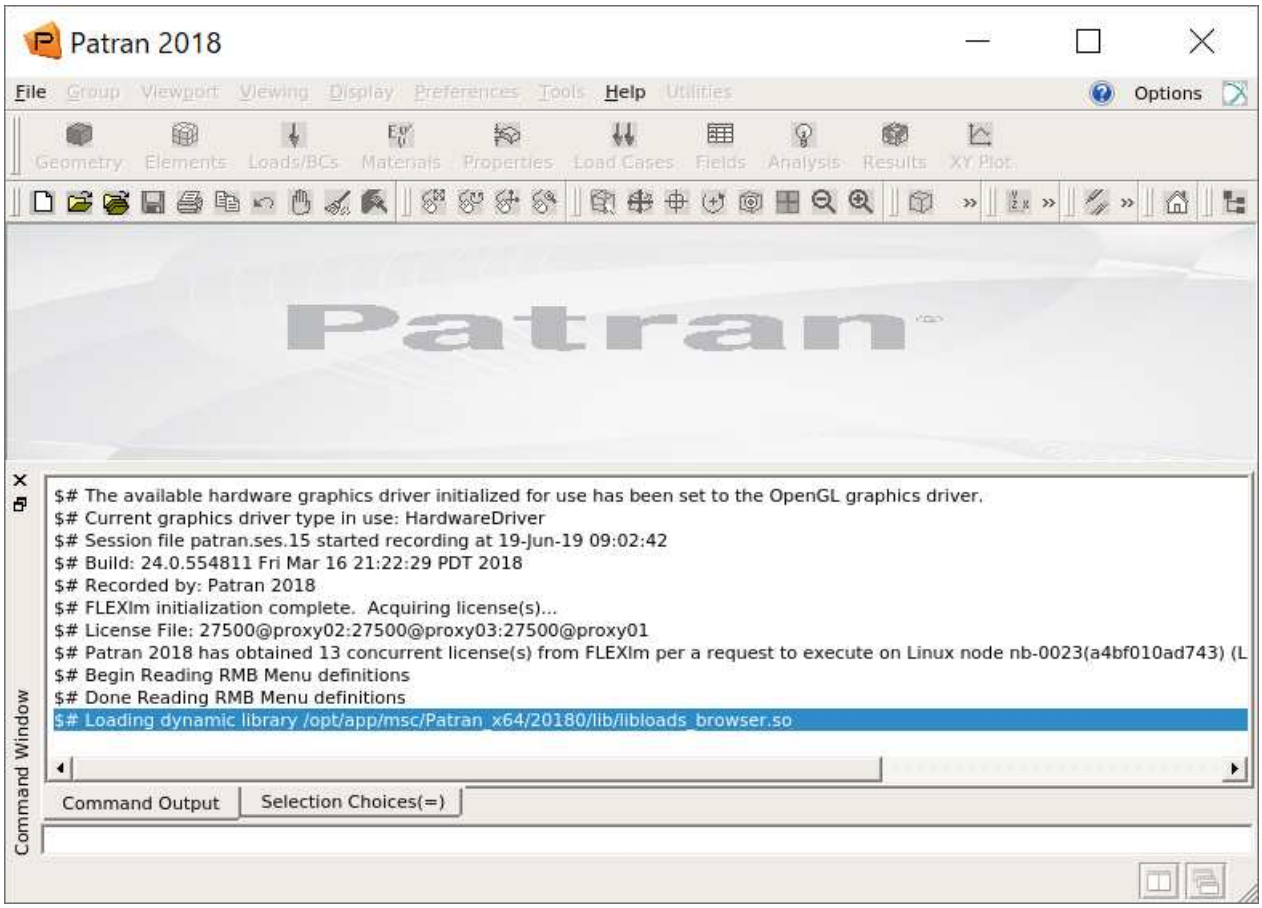
```
$ module load patran
```

Patranの環境設定を行うことにより、スーパーコンピュータシステムで提供しているジョブ管理用のコマンド **qstat** および **qkill** が、Patranのコマンドと競合します。これら2つのジョブ管理用コマンドについては、 **jobstat** および **jobkill** という名称でも提供していますので、別名の方をご利用ください。

起動方法

次のコマンドを実行するとPatranが起動します。 **tssrun** コマンドの詳細は 会話型処理 をご覧ください。

```
$ tssrun --x11 patran
```



メニューバー の **File** 内にある **Quit** をクリックするとPatranが終了します。

スレッド並列計算の利用

※パーソナルコースキューやグループコースキューをお持ちの方は、並列計算をご利用の際は前述のキューでアプリをご利用ください。 tssrun コマンドを実行の際に、 --rsc オプションでスレッド並列数を指定します。

例1：4並列実行... t,cの値を4に指定します

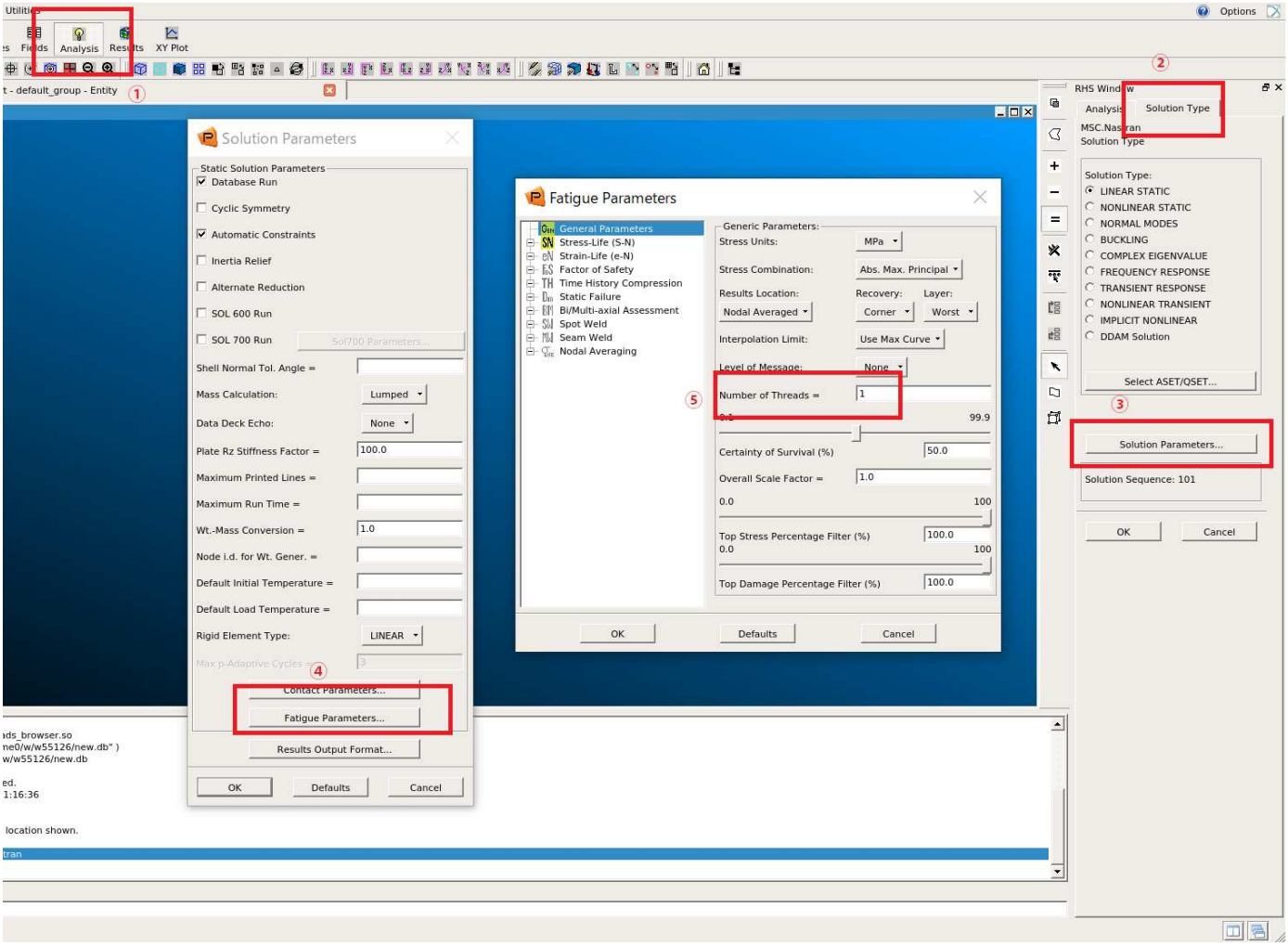
```
$ tssrun --x11 --rsc t=4:c=4 patran
```

例2：利用可能メモリも増量する場合... mの値も指定します（何も指定しないと各システムのデフォルト値となります）

```
$ tssrun --x11 --rsc t=4:c=4:m=10G patran
```

GUIの起動後、解析の実行前に下記の設定を行って、並列数を指定してください。なお、並列計算は疲労計算過程に対してのみ有効です。このため、疲労計算を行わない場合（計算結果の出力要求として疲労寿命を選択していない場合）は設定は無視されます。

[Analysis]→[Solution Type]→[Solution Parameters]→[Fatigue Parameters]→[Number of Threads] （4並列で実行したければ4を入力）



参考資料

日本語

Patran ドキュメント （MSC Software）

英語

Patran Docs （MSC Software）

リンク

外部リンク

Patran （MSC Software）

MSC Marc

利用環境

利用できるバージョン・システム

Modulesソフトウェアパッケージは、アプリケーションの利用に必要な環境設定をmoduleコマンドを実行することで、動的に切り替えて設定することができます。また、異なるバージョンのアプリケーションを切り替えて利用する際に、簡単に環境設定を変更することができます。詳細は Modules をご覧ください。

module avail コマンドにて，利用できるモジュールファイル一覧が確認できます．

バージョン	モジュールファイル名	システムA	システムB/C/G	クラウドシステム
2023.4 (default)	marc/2023.4	-	AU	-
2022.3	marc/2022.3	-	AU	-

+ ： すべてのユーザが利用可能
AU ： 学術研究機関限定で利用可能
- ： 利用不可

利用可能なライセンス数

同時に利用可能な並列数/ユーザ数には上限があります。 ライセンスの不足によるエラーが生じた場合は、ライセンスの利用に関してご協力を依頼することがあります。

機能概要

MSC.Marcは有限要素法による非線形汎用構造解析プログラムです。構造解析、熱伝導解析、音響解析、静電場解析などの解析処理を行なうことが出来ます。また、専用の会話型プリ・ポストプロセッサである Marc Mentat を利用すると、有限要素モデルの作成および解析結果の表示が可能です。

機能

Marcは、以下のようなライブラリで構成されており、これらを組み合わせることにより、様々な解析を行うことができます。

- 解析ライブラリ
  - 構造解析ライブラリ
  - 非構造解析ライブラリ
  - 連成構造ライブラリ
- 要素ライブラリ
- 材料ライブラリ
- 機能ライブラリ

利用分野

- 航空・宇宙
- 重工業
- 自動車
- 電気・電子
- 建設
- 医療関係　など

利用方法

環境設定(moduleコマンドの実行)

moduleコマンドを実行し、環境設定を行います。（利用したいバージョンのmoduleファイルをロードします）

```
$ module load marc
```

会話型での実行

形式

```
$ tssrun run_marc -j 入力ファイル -b no （オプション指定）
```

例　会話型(フォアグラウンド)で実行

```
$ tssrun run_marc -j sample.dat  -b no
```

例 スレッド数4 で並列実行

スレッド並列で実行する場合、 **-nthread** オプションで並列数を指定した上で、 **tssrun** コマンドを使用する必要があります。 **tssrun** コマンドの詳細は 会話型処理 をご覧ください。

```
$ tssrun --rsc t=4:c=4:m=20G run_marc -j sample -b no -nthread 4
```

バッチでの実行

バッチスクリプト内で、 **-j** オプションでの入力ファイルの指定に加えて、 **-b no -v n** オプションを指定してください。
バッチの利用方法は バッチ処理 をご覧ください。

バッチスクリプトの例 (SMP 4並列)

```
#!/bin/bash
#============ LSF Options ============
#SBATCH -p gr19999b
#SBATCH -t 1:00:00
#SBATCH --rsc p=1:t=4:c=4:m=20G
#============ Shell Script ============
module load marc
srun run_marc -j sample -b no -v n -nthread $OMP_NUM_THREADS
```

指定可能オプション

オプション	説明
-j jid	入力ファイル(*.dat)の指定
-b no	フォアグラウンド実行の指定
-v {y,n}	入力ファイルチェック
-nthread num	スレッド並列数
-cpu sec	CPU制限時間
-prog progname	前回のジョブで実行した実行可能プログラム “progname.marc” を実行
-user username	ユーザーサブルーチン “username.f” を使用して、新しい実行可能プログラム “username.marc” を生成

参考資料

日本語

Marc & Mentat Docs （MSC Software）

英語

Marc & Mentat Docs （MSC Software）

リンク

外部リンク

Marc （MSC Software）

講習会

プログラム講習会 を定期的に開催しています。講習会資料を配布していますので、ご利用ください。

MSC Marc Mentat

利用環境

利用できるバージョン・システム

アプリケーションの利用に必要な環境設定をmoduleコマンドを実行することで、動的に切り替えて設定することができます。 また、異なるバージョンのアプリケーションを切り替えて利用する際に、簡単に環境設定を変更することができます。詳細は Modules をご覧ください。

module avail コマンドにて、利用できるモジュールファイル一覧が確認できます。

バージョン	モジュールファイル名	システムA	システムB/C/G	クラウドシステム
2023.4 (default)	marc/2023.4	-	AU	-
2022.3	marc/2022.3	-	AU	-

+ ： すべてのユーザが利用可能
AU ： 学術研究機関限定で利用可能
KU ： 京都大学構成員限定で利用可能
- ： 利用不可

利用可能なライセンス数

同時に利用可能な並列数/ユーザ数には上限があります。 ライセンスの不足によるエラーが生じた場合は、ライセンスの利用に関してご協力を依頼することがあります。

機能概要

Msc.Marc MentatはMSC.Marc専用の会話型プリ・ポストプロセッサです。有限要素モデルの作成および解析結果の表示が可能です。

機能

Marc Mentatは対話型操作にて、Marcの入力データ作成・編集を行うことができます。また、Marcで解析した結果を表示することも可能です。

利用分野

- 航空・宇宙
- 重工業
- 自動車
- 電気・電子
- 建設
- 医療関係　など

利用方法

前提条件

Mentat はGUIアプリケーションであるため、X Windowの環境が必要です。

FastX、NICE DCVを使用してログインするか、Windowsでご利用いただけるX11 Forwarding に対応したSSHクライアントソフト(例えばMobaXterm)およびX11サーバをご利用いただき、GUIアプリケーションが起動可能な方法でシステムにログインしてください。

環境設定(moduleコマンドの実行)

moduleコマンドを実行し、環境設定を行います。（利用したいバージョンのmoduleファイルをロードします）

```
$ module load marc
```

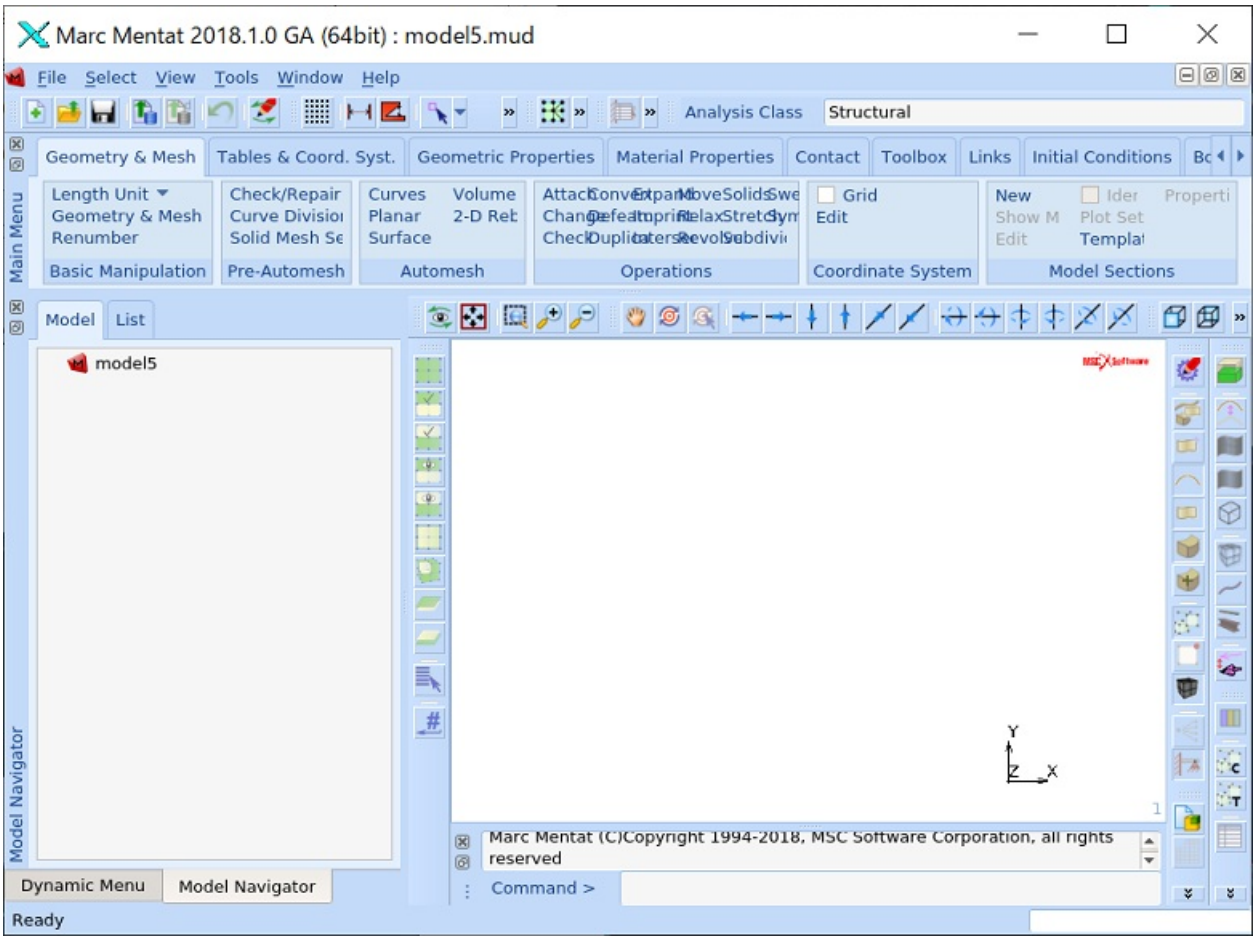
起動方法

次のコマンドを実行するとMarc Mentatが起動します。 **tssrun** コマンドの詳細は 会話型処理 をご覧ください。

```
$ tssrun --x11 mentat
```

日本語で起動する場合は、次のコマンドを実行します。

```
$ tssrun --x11 mentat -lang ja
```



メニューバー の **File** 内にある **Quit** をクリックし、 **SAVE & QUIT** で **Yes** を選択するとMarc Mentatが終了します。

スレッド並列計算の利用

※パーソナルコースキューやグループコースキューをお持ちの方は、並列計算をご利用の際は前述のキューでアプリをご利用ください。

まず、tssrunコマンドを実行の際に、--rsc オプションでスレッド並列数を指定します。

例1：4並列実行... t,cの値を4に指定します

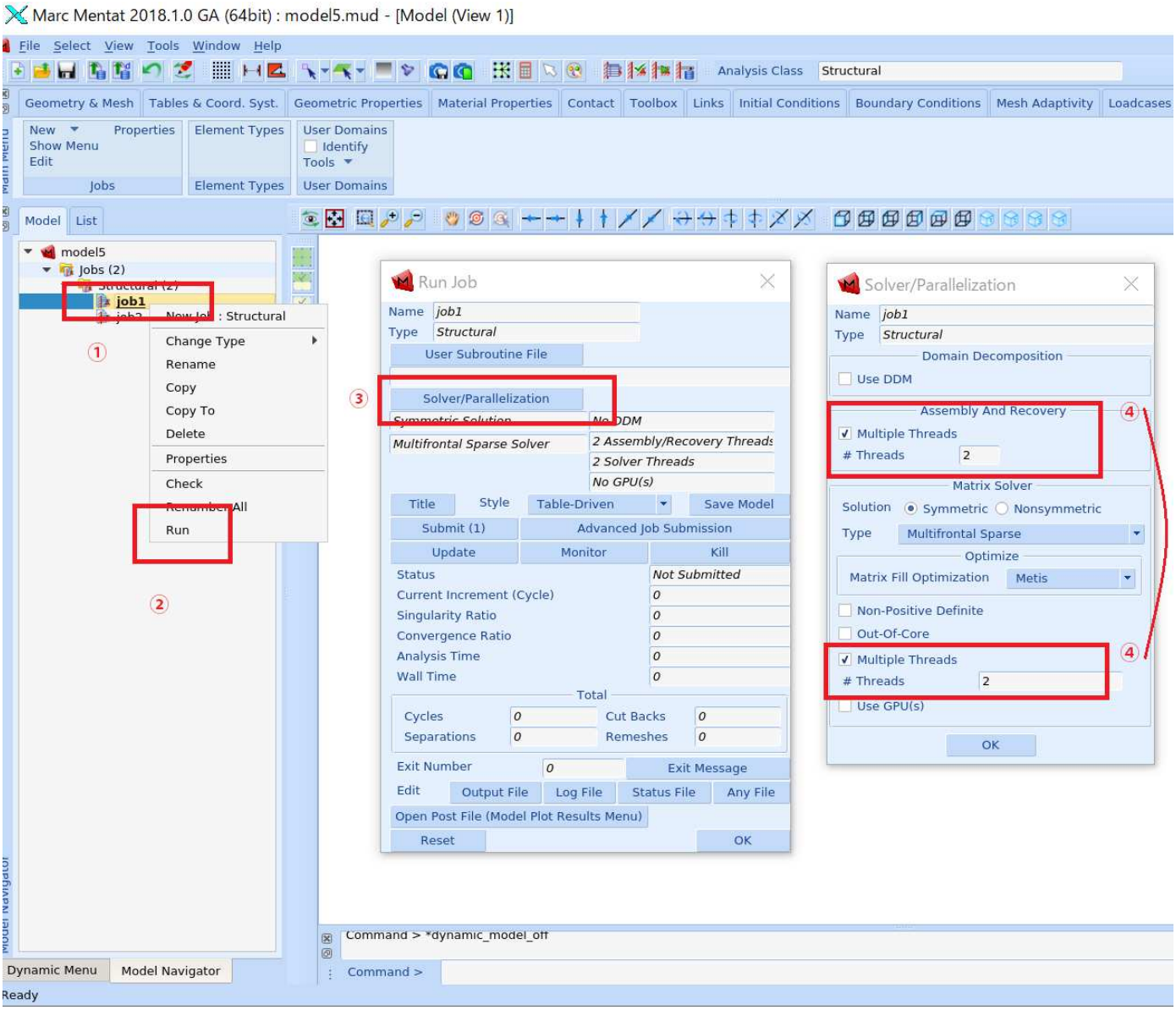
```
$ tssrun --rsc t=4:c=4 mentat
```

例2：利用可能メモリも増量する場合... mの値も指定します（何も指定しないと3413MBとなります）

```
$ tssrun --x11 --rsc t=4:c=4:m=10G mentat
```

GUIの起動後、解析の実行前に下記の設定を行って、並列数を指定してください。

[job名を右クリック]→[Run]→[Solver/Parallelization]→[Multiple Threads(2か所)] （4並列で実行したければ4を入力）



なお、2か所のMultiple Threadsのうち、「Matrix Solver」の方（図の下の方）では、次のソルバータイプで並列計算が有効になります。対応していないソルバータイプを指定すると、当該Multiple Threadsの指定ができません。

- Multifrontal Sparse
- Casi Iterative
- Paradiso Direct Sparse
- Mumps Parallel Direct

参考資料

日本語

Marc & Mentat ドキュメント （MSC Software）

英語

Marc & Mentat Docs （MSC Software）

リンク

外部リンク

Marc （MSC Software）

LS-DYNA

利用環境

利用できるバージョン・システム

Modulesソフトウェアパッケージは、アプリケーションの利用に必要な環境設定をmoduleコマンドを実行することで、動的に切り替えて設定することができます。また、異なるバージョンのアプリケーションを切り替えて利用する際に、簡単に環境設定を変更することができます。詳細は Modules をご覧ください。

module avail コマンドにて、利用できるモジュールファイル一覧が確認できます。

バージョン	モジュールファイル名	システムA	システムB/C/G	クラウドシステム
R14.1.0 (default)	ls-dyna/R14.1.0	-	AU	-
R13.1.1	ls-dyna/R13.1.1	-	AU	-

+ ： すべてのユーザが利用可能
AU ： 学術研究機関限定で利用可能
- ： 利用不可

一人で同時に利用可能なライセンス数

同時に利用可能なプロセス数は8ライセンスです。

利用状況によってはユーザ当たりの同時利用可能なライセンス数を制限する場合があります。

プリポストソフトウェア

LS-DYNAの入出力データを扱うプリポストソフトウェアとして、LS-PREPOSTがあります。

利用できるバージョン・システム

バージョン	モジュールファイル名	システムA	システムB/C/G	クラウドシステム
ls-prepost 4.10.5 (default)	ls-prepost/4.10.5	-	AU	-
ls-prepost 4.9.10	ls-prepost/4.9.10	-	AU	-

+ ： すべてのユーザが利用可能
AU ： 学術研究機関限定で利用可能
- ： 利用不可

利用方法

moduleコマンドを実行し、環境設定を行います（利用したいバージョンのmoduleファイルをロードします）。 moduleコマンドの詳細を知りたい方は Modules をご覧ください。

```
$ module load ls-prepost
```

FastX、NiceDCV など、X Window Systemが利用可能な環境でログインした後に、次のコマンドを実行すると起動します。 **tssrun** コマンドの詳細は 会話型処理 をご覧ください。

```
$ tssrun --x11 lspp49
```

なおLS-DYNA、LS-PREPOSTの使用方法の参考資料として、メディアセンター北館窓口で紙媒体の講習会資料を先着順で配布しています。（部数に限りがありますので、ご了承ください。）

機能概要

非線形動的構造解析ソフトウエアLS-DYNAは、陽解法により構造物の大変形、弾塑性、動的接触・挙動を時間履歴でシミュレーションするソフトウエアです。
LS-DYNAは、米国のローレンス・リバモア国立研究所（LLNL）の開発したDYNA3Dを改良したもので、Ver.950から標準機能として陰解法による構造解析や流体解析／熱解析も可能となりました。
陽解法・陰解法の搭載により、通常の線形解析や周波数応答にも対応し、ハイエンド汎用構造解析ソフトウェアとして活用できます。

機能

- 現象解析：　下記のような現象に対する解析を行えます。
  - 衝突／衝撃（自動車の衝突等）
  - 塑性加工（プレス成形等）
  - 落下（携帯製品の落下等）
  - 亀裂／破壊（ひび割れ等）
  - 歪を伴う機構解析（ゴムローラーによる搬送等）
- 接触
  LS-DYNAの接触解析は、20を超えるオプションが有効であり、接触面については固着、剥離、接触、滑り、摩擦などの条件が設定できます。
- 材料モデル
  100を超す金属、非金属材料モデルが用意されており、材料モデル毎にバネ、ダンパー、剛体、溶接などのさまざまな要素が用意されています。
- インターフェース
  CADデータや他の構造解析ソフト、その他のシミュレーションソフトとのインターフェースが用意されています。

利用分野

- 自動車
- 航空宇宙
- 防衛
- 電気機器
- 建設・土木

- 原子力　など

利用方法

環境設定(moduleコマンドの実行)

moduleコマンドを実行し、環境設定を行います。（利用したいバージョンのmoduleファイルをロードします）

```
$ module load ls-dyna

## MPP版を利用する場合
$ module load intelmpi/2018.4
```

使用可能なコマンド

コマンド名	種別	備考
lsdyna_sp	SMP版、単精度	
lsdyna_dp	SMP版、倍精度	以下の利用方法はこのコマンドを使用して解説
lsdyna_sp_mpp	MPP版、単精度	intelmpi/2018.4 のモジュールロードが必須
lsdyna_dp_mpp	MPP版、倍精度	intelmpi/2018.4 のモジュールロードが必須

会話型での実行

形式

```
$ tssrun lsdyna_dp i=[実行ファイル]  ncpu=[並列CPU数(指定しない時は"1")]
```

実行ファイル、並列CPU数以外にもオプションの指定が可能です。詳細はManual (LS-DYNA KEYWORD MANUAL)をご覧ください。

例 2CPUの並列処理を行う

```
$ tssrun lsdyna_dp i=sample.dyn ncpu=2
**tssrun** コマンドの詳細は [会話型処理](/run/interactive) をご覧ください。
```

出力ファイル

```
「d3hsp」…解析計算履歴情報出力ファイル
「messag」…計算実行メッセージ、エラーメッセージ出力ファイル
「status.out」…解析結果出力ファイル
「d3plot」…モデルの３次元形状出力ファイル
「d3dump」…リスタート実行時必要データファイル
```

バッチジョブでの実行

バッチジョブで実行するためには、バッチファイルを用意する必要があります。スクリプト内でコマンドラインで実行するのと同様にコマンドを記述し、 **sbatch** コマンドでジョブを投入します。

```
#!/bin/bash
#============ Slurm Options ============
#SBATCH -p gr19999b
#SBATCH -t 1:00:00
#SBATCH --rsc p=1:t=2:c=2:m=2G
#============ Shell Script ============

module load ls-dyna

srun lsdyna_dp i=sample.dyn ncpu=$SLURM_DPC_CPUS
```

参考資料

- LSDYNA Manuals (Ansys)
- LSDYNA Relase Note R13.0.0
  - English
  - Japanese
- LSDYNA Relase Note R13.1.0
  - English
  - Japanese
- LSDYNA Relase Note R13.1.1
  - English
  - Japanese
  - 不具合・仕様変更情報

リンク

外部リンク

- Ansys LS-DYNA （株式会社JSOL）

ANSYS

利用環境

利用できるバージョン・システム・ライセンス

Modulesソフトウェアパッケージは、アプリケーションの利用に必要な環境設定をmoduleコマンドを実行することで、動的に切り替えて設定することができます。また、異なるバージョンのアプリケーションを切り替えて利用する際に、簡単に環境設定を変更することができます。詳細は Modules をご覧ください。

module avail コマンドにて、利用できるモジュールファイル一覧が確認できます。

バージョン	モジュールファイル名	システムA	システムB/C/G	クラウドシステム
2024R1.1 (default)	ansysmcfd/2024R1.1	-	KU	-
2023R1.4	ansysmcfd/2023R1.4	-	KU	-
2022R2.2	ansysmcfd/2022R2.2	-	KU	-
2022R2	ansysmcfd/2022R2	-	KU	-

+ ： すべてのユーザが利用可能
KU ： 京都大学構成員限定で利用可能
- ： 利用不可

提供ライセンス

- ANSYS Academic Research Mechanical and CFD
  - ランチャーで選択するライセンスは、「Ansys Mechanical Enterprise」としてください。
  - 利用可能な機能は下のサイトからご確認ください
    Product Features

機能概要

ANSYSはマルチフィジックスCAEです。構造・振動・伝熱・電磁場・圧電・音響・熱流体・落下衝突などの物理現象や、それらを組み合わせた連成問題を、目的に合わせて柔軟に解析することが可能です。

機能

- 構造解析
- 伝熱解析
- 電磁場解析
- 流体解析
- 疲労解析
- 音響解析
- 落下・衝突解析
- 圧電解析

- 連成解析
- FSI 解析（流体-ソリッド連成解析）

利用分野

- 自動車
- 電気電子
- 航空宇宙
- エネルギー
- 医療工学
- 建築・土木　など

利用方法

利用登録

ANSYSを利用するには、 利用者ポータル から申請いただく必要があります。

※ANSYSを利用できるのは、京都大学構成員、かつ吉田本部構内から**50**マイル（約**80.4**キロ）以内に物理的に居る方に限ります。

環境設定(moduleコマンドの実行)

moduleコマンドを実行し、環境設定を行います。（利用したいバージョンのmoduleファイルをロードします）

```
$ module load ansysmcfd
```

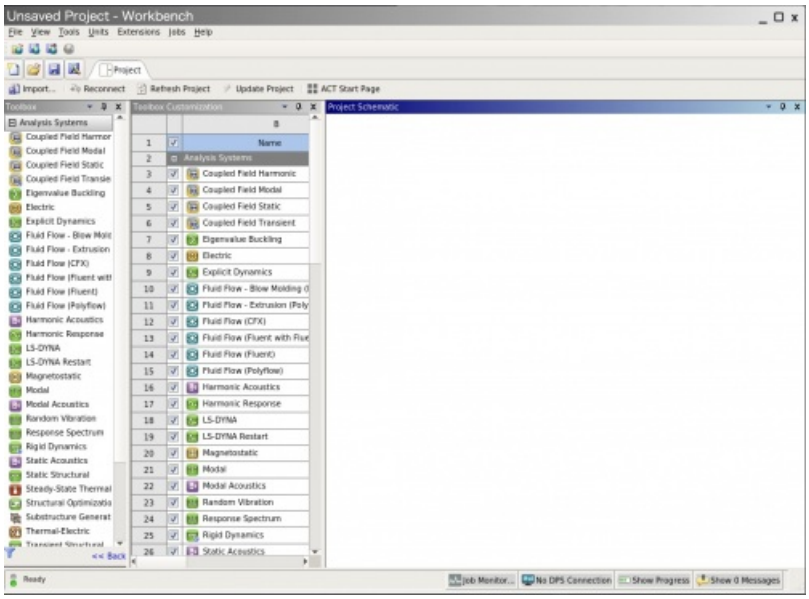
GUIでの利用方法

ANSYS Workbenchの起動

以下のコマンドを実行すると、ANSYS Workbenchを起動することができます。 ランチャー経由で各種プロダクトを起動する場合は、ANSYS Launcherの起動を ご参照下さい。

なお、参考資料としてセミナー資料を掲載しておりますので、必要に応じてご活用ください。

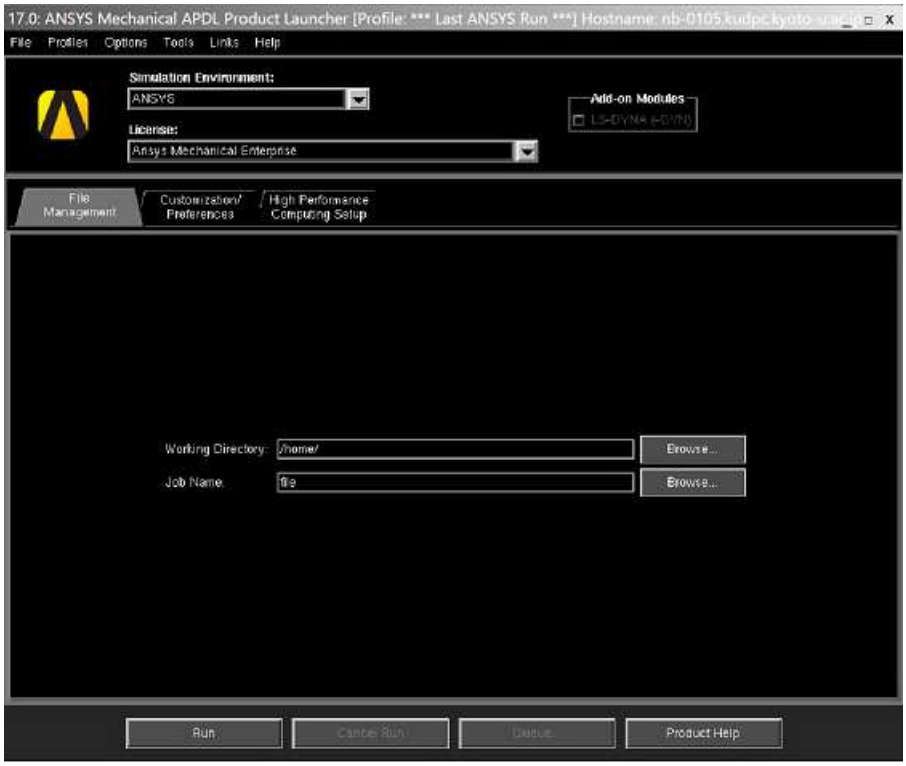
```
$ tssrun --x11 runwb2
```



ANSYS Launcherの起動

以下のコマンドを実行すると、ANSYS Launcherが起動します。 ANSYS Launcher経由で各種プロダクトを起動することが可能です。 **tssrun** コマンドの詳細は 会話型処理 をご覧ください。

```
$ tssrun --x11 launcher
```
※バージョンにより実行コマンド名の語尾(バージョン番号)が異なります。



シミュレーションで並列計算を利用したい場合は、tssrun コマンドを実行する際、確保するコア数を --rsc オプションで指定してください。

例：4並列計算

```
$ tssrun --x11 --rsc t=4:c=4 launcher170
```

なお、並列計算をご利用の場合は、下記で述べる「シミュレーション環境の設定」において、「High Performance Computing Setup」タブ内でUse Shared-Memory Parallelの値も設定する必要があります。

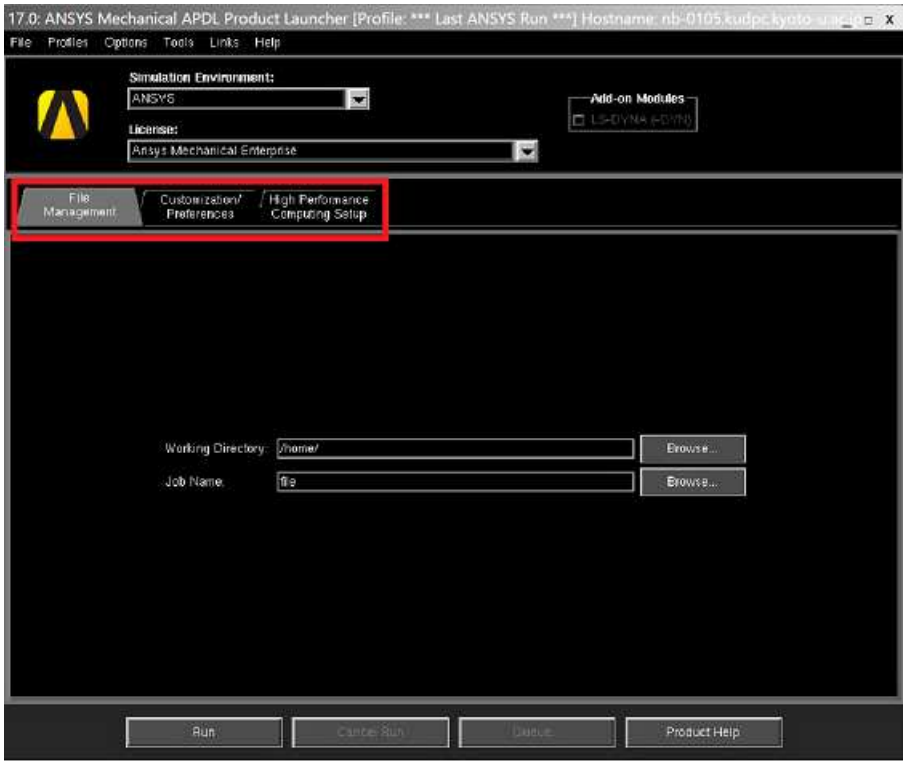
シミュレーション環境の設定

ランチャーが起動したら、シミュレーション環境の設定を行います。

シミュレーション環境(Simulation Enviroment)として、以下の3つが選択できます。ここでは、ANSYSを選択した場合について解説を進めていきます。

- ANSYS … ANSYS標準インターフェイス
- ANSYS Batch … 解析をバッチ処理する場合に使用

シミュレーション環境ごとの設定は、タブメニューより行います。



ファイル管理タブ **(File Management)**

- ワーキングディレクトリ(Working Directory)
  - ANSYS実行中に作成されるデフォルトのファイル保存先
- ジョブネーム(Job Name)
  - ANSYSを実行中に作成されるファイルのデフォルトファイル名。最大32文字

カスタマイゼーション/プリファレンスタブ **(Customization/Preferences)**

- メモリ
  - 解析中のメモリは自動で設定されるが、任意の値を設定することもできる

- 言語選択(Linux/UNIXでは英語のみ)
- グラフィックデバイス
  - X11 … デフォルト
  - X11c
  - 3D … openglに対応している必要がある
- 出力先
  - Screen & File … デフォルト
  - File Only ... バージョン2021R2, 2022R1を使う場合、File Onlyを選択してください。
  - Screen Only

HPCセットアップ (High Performance Computing Setup)

- 実行方法(Type of High Perfomance Computing Run)
  - None … シリアル実行
  - Use Shared-Memory Parallel (SMP) … スレッド並列実行
    - tssrun コマンドの --rsc オプションで指定した並列数と同じ数字を設定してください。16並列まで利用できます。

ANSYSの起動

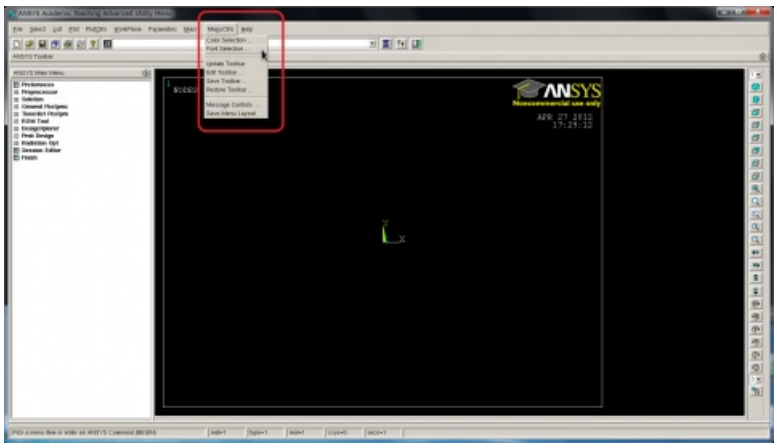
各種設定が終わったらRunボタンをクリックすることでANSYS（選択したシミュレーション環境）が起動します。

ANSYS起動後の設定

フォントサイズの変更:

デフォルトのメニューのフォントサイズが小さい（あるいは大きい）場合、フォントサイズを変更することが可能です。以下のメニューで変更することができます。

[MenuCtrls] ⇒ [Font Selection]



CUIでの利用方法

ANSYSは標準でGUIを利用して会話形式で処理を進めますが、コマンドラインで実行することも可能です。以下にコマンドシンタックスとオプションを記載しています。 **tssrun** コマンドの詳細は 会話型処理 をご覧ください。

```
$ tssrun ansys222 [options] -i input-file > output-file
```
※バージョンにより実行コマンド名の語尾(バージョン番号)が異なります。

オプション	概要
-j JobName	ジョブ名を指定できます。
-d Switch	デバイスタイプを指定します。 X11, X11c, 3D
-m Size	メモリの指定：ワークスペース領域(単位:MB) 標準は自動設定
-db Size	メモリの指定：データベース領域(単位:MB) 標準は自動設定
-p Prod	プロダクトの指定
-g	GUI モードで起動
-np NCPUS	プロセス並列数 (-smp 指定時または -dis 未指定時)
-np NTHREAD	スレッド並列数 (-dis 指定時)
-smp	SMP 並列
-dis	MPI 並列
-dir Dir	作業ディレクトリの指定

バッチジョブでの利用方法

バッチジョブで実行するためには、バッチファイルを用意する必要があります。
スクリプト内でコマンドラインで実行するのと同様にコマンドを記述し、sbatch コマンドでジョブを投入します。

ジョブスクリプトの例1: SMP

```
#!/bin/bash
#============ Slurm Options ============
#SBATCH -q gr19999b
#SBATCH -t 1:00:00
#SBATCH --rsc p=1:t=4:c=4
#============ Shell Script ============
module load ansysmcfd/2022R2
ansys222 -j test_${SLURM_JOBID} -b -smp -np ${SLURM_DPC_THREADS} -i test.dat
```

※ ANSYSの実行コマンドは、コマンドの内部で mpiexec を自動で呼び出すので、srun コマンドは不要です。
※ ANSYSのバージョンにより実行コマンド名の語尾(バージョン番号)が異なります。

ジョブスクリプトの例2: MPI

```
#!/bin/bash
#============ Slurm Options ============
#SBATCH -q gr19999b
#SBATCH -t 1:00:00
#SBATCH --rsc p=4:t=1:c=1
#============ Shell Script ============
module load ansys
ansys222 -j test_${SLURM_JOBID} -b -dis -np ${SLURM_DPC_NPROCS} -i test.dat
```

※ ANSYSの実行コマンドは、コマンドの内部で mpiexec を自動で呼び出すので、srun コマンドは不要です。
※ ANSYSのバージョンにより実行コマンド名の語尾(バージョン番号)が異なります。

ジョブスクリプトの例3: ANSYS Fluent

処理リスト化したファイル(journal.jou)を用意し、3dの倍精度ソルバーで解析を実行する例です。

```
#!/bin/bash
#============ Slurm Options ============
#SBATCH -q gr19999b
#SBATCH -t 1:00:00
#SBATCH --rsc p=1:t=4:c=4
#============ Shell Script ============
module load ansysmcfd/2022R2

srun fluent -g 3ddp -i journal.jou
```

参考資料

日本語

2022R1

- ANSYS Workbench Mechanical 入門セミナー
- ANSYS Workbench Mechanical 使いこなしセミナー
- ANSYS Workbench Mechanical 接触解析セミナー
- ANSYS Workbench Mechanical 動解析セミナー
- ANSYS Workbench Mechanical 材料非線形セミナー
- ANSYS Workbench Mechanical 伝熱解析セミナー
- ANSYS Fluent ベーシックセミナー
- ANSYS CFX ベーシックセミナー

外部リンク

- ANSYS, inc.
- アンシス・ジャパン株式会社
- 汎用FEM連成解析ツール ANSYS：サイバネット

Jupyter Lab

Jupyter Labを計算ノードで実行する場合は、以下のような方法で起動することができます。 なお、プロセス並列の実行には対応していません。

計算ノードでの起動方法

anaconda3に同梱されている JupyterLabの使い方を示します。

ユーザ名**(b59999)**、キュー名**(gr19999b)**は適宜変更してください。

1. モジュールをロードします

```
$ module load anaconda3
```

2. JupyterLabを計算ノードで起動します。（例：システムBでgr19999b キューを使う場合）

- バッチ処理の場合
  ジョブスクリプトを作成します。

```
$ vi run.sh
#!/bin/bash
#============ Slurm Options ===========
#SBATCH -p gr19999b
#SBATCH --rsc c=8               # 要求リソースの指定(8コアの例)
#SBATCH -t 3:00:00
#SBATCH -o %x.%j.out            # ジョブの標準出力ファイルの指定
#============ Shell Script ============
set -x
srun jupyter lab --no-browser --ip=0.0.0.0
```

  ジョブを投入します。

```
$ sbatch run.sh
```

- 会話型処理の場合
  ジョブを投入します。

```
$ tssrun -p gr19999b --rsc c=8 jupyter lab --no-browser --ip=0.0.0.0
```

3. Jupyter Labが計算ノードで起動すると、以下のような接続情報が表示されます。

- バッチ処理の場合

```
[b59999@laurel31 ~]$ tail <標準出力ファイル>
[C 2024-05-24 15:34:18.069 ServerApp]

To access the server, open this file in a browser:
    file:///home/b/b59999/.local/share/jupyter/runtime/jpserver-4144334-open.html
Or copy and paste one of these URLs:
    http://xb0037:8888/lab?token=f1031ba47cc702ccaac0433646ed4a848827a88b14521841
    http://127.0.0.1:8888/lab?token=f1031ba47cc702ccaac0433646ed4a848827a88b14521841
```

- 会話型処理の場合

```
[b59999@laurel31 ~]$ tssrun -p gr19999b jupyter lab --no-browser --ip=0.0.0.0
（省略）
[C 2024-05-22 17:08:38.648 ServerApp]

To access the server, open this file in a browser:
    file:///home/b/b59999/.local/share/jupyter/runtime/jpserver-4144334-open.html
Or copy and paste one of these URLs:
    http://xb0037:8888/lab?token=f1031ba47cc702ccaac0433646ed4a848827a88b14521841
    http://127.0.0.1:8888/lab?token=f1031ba47cc702ccaac0433646ed4a848827a88b14521841
```

起動したJupyterLabへの接続方法

FastXを用いる場合

1. ログインノード上でWebブラウザを起動します。

```
$ firefox
```

2. 計算ノードでの起動方法の項番3で取得した接続情報のうち、下から2行目に記載の 「http://xbXXXX:ポート番号/lab?token(以下略)」から始まる行の 「xbXXXX」を「xbXXXX-ib0」に変更し、アドレス欄に入力して接続します。
（例）http://xb0037-ib0:8888/lab?token=f1031ba47cc702ccaac0433646ed4a848827a88b14521841

コマンドプロンプトやターミナルなどを用いる場合

1. 別ターミナルを起動し、ポートフォワーディングでシステムに接続します。（例：システムBにb59999で接続する場合）

```
$ ssh -L ポート番号:xbXXXX-ib0:ポート番号 b59999@laurel.kudpc.kyoto-u.ac.jp
```

計算ノードでの接続方法の項番3で取得した接続情報のうち、 下から2行目に記載されている、「http://xbXXXX:

ポート番号/lab?token(以下略)」 の「xbXXXX」を「xbXXXX-ib0」に変更し、ポート番号は「xbXXXX」に続く4桁の数字を入力してください。
（例）$ ssh -L 8888:xb0037-ib0:8888 b59999@laurel.kudpc.kyoto-u.ac.jp

2. コマンドプロンプトやターミナルを閉じずに、Webブラウザでの表示 へお進み下さい。

MobaXtermを用いる場合(Windowsのみ)

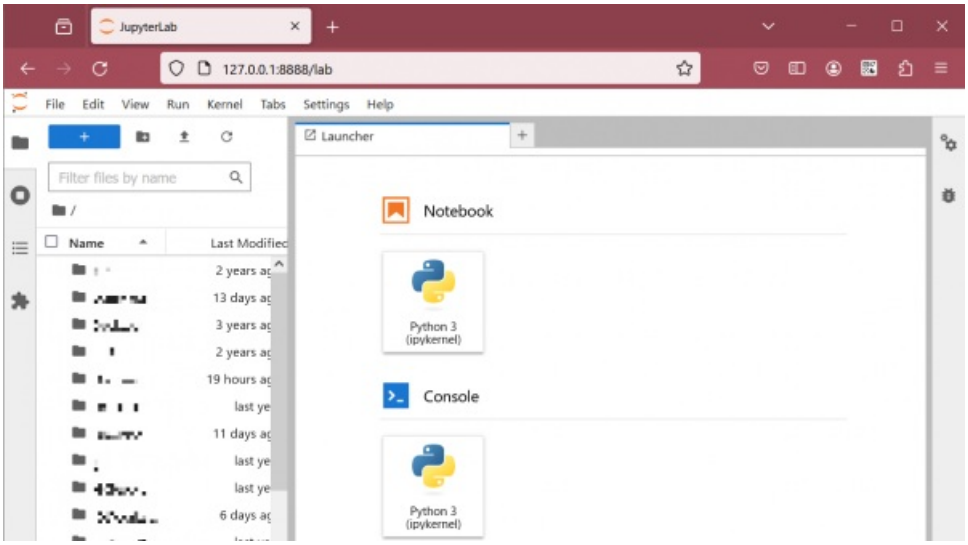
1. ポートフォワーディングの設定 を参考に、ポートフォワーディングの設定をします。 （例：システムBにb59999で接続する場合の手順4. の設定）
    - <Forwarded port>：ポート番号
    - <SSH server>：laurel.kudpc.kyoto-u.ac.jp
    - <SSH login>：b59999
    - <SSH port>：22
    - <Remote server>：xbXXXX-ib0
    - <Remote port>：ポート番号

計算ノードでの接続方法の項番3で取得した接続情報のうち、 下から2行目に記載されている、「http://xbXXXX:ポート番号/lab?token(以下略)」 の「xbXXXX」を「xbXXXX-ib0」に変更し、ポート番号は「xbXXXX」に続く4桁の数字を入力してください。
（例）<Forwarded port>：8888、<Remote server>：xb0037-ib0、<Remote port>：8888

手順7. まで終了したら、Webブラウザでの表示 へお進み下さい。

Webブラウザでの表示

1. お手元のコンピュータ上で、Webブラウザを立ち上げます。
2. Webブラウザのアドレス欄に、計算ノードでの接続方法の項番3で 出力された接続情報のうち、一番下の行に記載されているURL(http://127.0.0.1:ポート番号/lab?token(以下略)) を最後までコピーし、貼り付けます。
3. 以下のような画面が表示されたら接続成功です。



終了方法

JupyterLabを起動したプロンプトでCtrl+C を2回押して、JupyterLabを終了します。

各種ツールのインストール

MobaXterm

MobaXtermはSSHクライアント、X11サーバ、ネットワークツール等を備えたWindows用拡張ターミナルです。MobaXtermのホームページからダウンロードすることができます。
インストール方法については、MobaXtermのインストールをご覧ください。

FastX

FastXは、HTTPS(443/TCP)を用いて、スーパーコンピュータシステムに 接続することが可能な X Window Systemです。
インストール方法については、FastXのインストールをご覧ください。

Archaea

Archaea(旧HCP Tools)は、HpFPというプロトコルを使用したファイル転送ツールです。専用のパッケージをインストールしていただくことでご利用いただけます。
インストール方法については、Archaeaのインストールをご覧ください。

MobaXterm

MobaXtermとは

MobaXtermは、SSHクライアント、X11サーバ、ネットワークツール等を備えたWindows用拡張ターミナルです。

インストール

MobaXtermのホームページ の「Download」より、Installer edition をダウンロードします。ダウンロードしたzipファイルを展開し、インストーラ (MobaXterm_installer_xx.msi) を実行します。※xxはバージョン番号です

Archaea

Archaea(旧HCP Tools)とは

高速なファイル転送ツールとして、Archaea(旧HCP Tools)を提供しています。

Archaea(旧HCP Tools)は、HpFPというプロトコルを使用しているため、高速なファイル転送が可能です。専用のパッケージをインストールしていただくことでご利用いただけます。

※「HCP Tools」は「Bytrix Archaea tools」に名称が変更されました。

クライアントソフトについて

バージョン

サーバにはArchaea tools ver1.4がインストールされています。クライアントソフトをインストールする際は **Archaea tools ver.1.4** もしくは **Archaea dialog ver.1.0** をインストールしてください。

対応OS

- Windows：10
- RHEL：7 / 8 / 9
- Ubuntu：1804 / 2004 / 2204
- Raspberry Pi OS：9 / 11
- openSUSE Leap：15.2 / 15.3 / 15.4
- macOS：12 / 13

クライアントソフトの入手

ダウンロード【最新版】（CLEALINK TECHNOLOGY）

インストール方法

ダウンロード【最新版】（CLEALINK TECHNOLOGY）のインストール方法をご覧ください。

使い方

Archaea(旧HCP Tools)の使い方は、Archaea(旧HCP Tools)によるファイル転送をご覧ください。

グループ管理者向け情報

グループ管理者とは

グループ管理者とは、グループコース、専用クラスタコースの管理を行う権限を持った人を指します。
初期設定では、グループ管理者はサービスコースの申請者となっています。また、サービスコースの申請者はグループ管理者を追加/変更することも可能です。追加/変更は利用者ポータル から申請してください。

グループ管理者は、グループを管理するためのコマンドを利用することができます。この管理者専用コマンドによって、グループメンバーやグループに割り当てられたキューやディスクを管理できます。

グループメンバーの管理

グループのメンバーは、利用者ポータル にログインすることで、メンバーの確認および追加/削除が可能です。

LARGEディスクスペースのバックアップ設定(group_backup)

group_backup コマンドで、LARGEディスクスペースのバックアップの設定ができます。LARGEディスクスペースは、*/LARGE0/グループ名* ディレクトリと */LARGE1/グループ名* ディレクトリで構成されており、以下の2つの状態のどちらかを設定できます。

設定状態	/**LARGE0**/グループ名	/**LARGE1**/グループ名
バックアップ使用	Safe（バックアップあり）	Backup （バックアップ先）
バックアップ未使用	Unsafe（バックアップなし）	Unsafe（バックアップなし）

このうち、設定がSafe または Unsafeとなっているディスクを利用できます。

バックアップ設定の確認

対象とするグループは -g オプションで指定可能です。省略した場合は、コマンド実行時のカレントグループが対象となります。

```
$ group_backup -g gr19999 -l
 Num  Filesystem             Status  Filesystem             Status
  1)  /LARGE0/gr19999  ... Safe    /LARGE1/gr19999    ... Backup  <- バックアップ使用状態
```

バックアップ未使用に設定

```
$ group_backup -g gr19999 --unsafe 1
/LARGE0/gr19999: Safe    => UnSafe
/LARGE1/gr19999: Backup => UnSafe
```

バックアップ設定の確認（変更後）

```
$ group_backup -g gr19999 -l
 Num  Filesystem            Status  Filesystem            Status
  1)  /LARGE0/gr19999    ... UnSafe  /LARGE1/gr19999    ... UnSafe  <- バックアップ未使用状態
```

バックアップ使用状態に戻す

```
$ group_backup -g gr19999 --safe 1
/LARGE0/gr19999: Unsafe => Safe
/LARGE1/gr19999: Unsafe => Backup
```

LARGEディスクスペースのファイル整理(group_trash)

group_trash コマンドで、LARGEディスクスペースにあるファイルを削除(ゴミ箱へ移動)することができます。卒業などでいなくなった利用者のファイルなど、ファイルオーナーが存在しないファイルなどを削除することができます。もし誤って削除してしまった場合はゴミ箱から復元できますが、ゴミ箱は毎週月曜日に空になりますので注意してください。

group_trashコマンドで削除

対象とするグループを -g オプションで指定してください。省略した場合は、コマンド実行時のカレントグループでグループ管理者の権限を判定します。

```
$ group_trash -g gr19999 /LARGE0/gr19999/file1
file1 to Trash (/LARGE0/gr19999/.DpcTrash/b59999/2009-04-10_1010)
```

キュー利用メンバーの管理

Slurmのキューの権限の管理の単位として、ユーザとグループの2種類があります。ユーザは初期設定では空にしており、 グループには初期設定としてキュー名に対応するグループが最初から登録されています。

複数のグループでキューを利用する場合や、グループに所属していない単一の利用者に対してキューの利用権限を付与したいと いったご要望がありましたら、問い合わせフォーム より お問い合わせ下さい。

キューのジョブスケジューリングポリシー

本センターのジョブスケジューリングポリシーは、以下の3種類から選択することができます。 個別キュー(grXXXXXx)をご契約いただいている場合は、 申請者の方から、問い合わせフォーム へ ご希望のスケジューリングポリシーをご連絡いただくことで、変更することが可能です。 (エントリコースやパーソナルコースなどの共有キューについては、ご希望をお受けすることはできません)

```
設定値　動作
```

設定値　動作	
pass	あるジョブを実行するのに十分な計算資源がある場合、そのジョブよりも前に並んでいる実行待ちジョブを追い越して実行する。 効率的に計算資源を利用できるが、大規模なジョブがいつまでも実行されない可能性が生じる。【初期設定値】
wait	計算資源に空きがある場合でも、ジョブ間の追越しは発生しない。
backfill	各ジョブの実行時間制限（-t）をもとに計算を行い、他のジョブの実行開始時刻に影響を及ぼさない場合のみ、追越しが発生する。たとえば、大規模ジョブが開始されるまでの間に実行を完了できる小さなジョブを走らせることで資源を有効活用できる。

キューのジョブ実行状況の確認とキャンセル(spadmin)

spadmin コマンドを用いることで、グループ管理者として登録されているキューのジョブ実行状況の確認、ジョブのキャンセルを行うことができます。

- ジョブの実行状況の確認

```
$ spadmin list -p gr19999b ## gr19999bの部分は確認したいキュー名に変更してください.
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
              4781  gr19999b run_cpu2   b59999  R    1:26:09      1 nb0001
```

- ジョブのキャンセル

```
$ spadmin cancel 123
scancel: Terminating job 123
```

なお、グループ管理者以外が **spadmin** コマンドを実行した場合は、以下のようなエラーが出力されます。

```
$ spadmin list -p gr19999g
Authorization Failure
```

HPCI利用に関する情報

このページでは、HPCIで京都大学学術情報メディアセンターを利用される方向けに利用方法を説明します。

利用開始手続きについて

利用登録を完了された方には、Eメールにて登録完了通知を送付いたします。 登録完了通知を受け取ったら、利用開始手続きを行ってください。

利用者番号の用途

「登録完了通知」でお知らせした利用者番号は、下表の用途に使用します。 なお、両方の区分に該当する場合もあります。 プライマリセンターならびに計算資源については、HPCI申請支援システムで確認できます。

区分	利用者番号の用途
京都大学をプライマリセンターに指定の方	証明書の発行など、WEB認証が必要な際に **HPCI**アカウント として使用します。

システムの利用方法

システムへのログイン方法

電子証明書の発行ならびにログインの手順は、HPCIから提供されているマニュアルをご覧ください。 京都大学の計算資源を使用する場合のホスト名は、以下の通りです。 なお、電子証明書発行後にログインを許可する登録処理を行いますので、15分程度時間を空けてログインしてください。

システム名	ホスト名
システムA	camphor.kudpc.kyoto-u.ac.jp

HPCIでは電子証明書を使用して、GSI認証 (Grid Security Infrastructure)によるSSH(GSI-SSH)により資源提供機関にログインします。

京都大学の計算資源経由でHPCIを利用する場合

京都大学の計算資源には、GSI-SSHに必要となる gsissh コマンドおよび myproxy-logon コマンドを用意してあります。

京都大学の計算資源を利用可能な方は こちら のページを参考にログインいただくと、 HPCI用の環境構築を行わなくとも、京都大学の計算資源経由で他のシステム構成機関にログインすることが可能です。 その場合は、以下のように myproxy-logon(代理証明書の取得)および gsissh コマンドを利用してください。

```
## 代理証明書の取得（hpci00XXXX は 自身のHPCI-IDに置き換え）
$ myproxy-logon -s portal.hpci.nii.ac.jp -l hpci00XXXX

## 他の資源提供機関へのログイン{#login_of_other_facility}
$ gsissh host01.example.jp
```

システムの利用方法

システムの利用方法は、システムへの接続方法 などをご覧ください。 HPCIで利用できるシステムはシステムAとなっています。

計算資源の利用

HPCIの計算資源を利用するには、バッチでのジョブ投入時に以下のキュー名を指定する必要があります。 バッチシステムの詳しい利用方法は、バッチ処理 をご覧ください。 キュー名に含まれる**HPCI課題ID**は初回採択時課題**ID**が利用されます。

分類	システム	種別	キュー名	ノード数(2024年度)	備考
HPCI	A	通期利用	hpa	106 ノード	HPCIの利用者で共有します。複数の課題で共有して使用しますので、長期間の占有利用は控えてください。
HPCI-JHPCN	A	通期利用	jha	26 ノード	HPCI-JHPCNの利用者で共有します。複数の課題で共有して使用しますので、長期間の占有利用は控えてください。
HPCI-JHPCN	A	集中利用	jhXXXXXXa	-	「jhXXXXXX」は課題IDに置き換えてください。利用期間は課題代表者に個別に通知します。

課題ID(グループ)の指定

京都大学では、HPCIで複数課題に採択された場合でも同じのログインID(利用者番号)を使用します。 バッチでのジョブ投入時に、キュー名およびグループ名を明示的に指定することで課題を切り替えてください。

下の例はバッチジョブの投入に必要となるジョブスクリプトのサンプルです。 「#SBATCH -p 」でキュー名にjhaを指定しています。

```
$ cat sample.sh
#!/bin/bash
#============ Slurm Options ============
#SBATCH -p jha
#SBATCH -t 2:00:00
#SBATCH --rsc p=4:t=8:c=8:m=1800M
#============ Shell Script ============
srun ./a.out
```

ストレージの利用

京都大学のストレージ

HPCIの課題で利用できる京都大学のストレージのパスは以下の通りです。 /LARGE0, /LARGE1 の組み合わせを利用いただけます。 利用可能なLARGE領域は課題代表者及び連絡責任者の方にメールで通知しています。

課題あたりに利用可能な容量は以下の通りです。

課題	利用可能な容量
HPCIのシステムA利用課題	資源提供通知の大容量ディスク欄に記載 (/LARGE0, /LARGE1 に半分ずつ割当)
HPCI-JHPCNのシステムA利用課題	資源提供通知の大容量ディスク欄に記載 (/LARGE0, /LARGE1 に半分ずつ割当)

/LARGE1は、初期状態で/LARGE0のバックアップ先となっています。 /LARGE1はバックアップを解除することで利用可能になります。 バックアップ設定の変更は、お問い合わせフォーム よりご依頼ください。

この他にホームディレクトリが、100GBまで利用できます。 ストレージの詳しい利用方法は、ファイルシステムの利用をご覧ください。

共用ストレージ

HPCI共用ストレージの利用が可能な課題の利用者向けのマウントポイントは以下の通りです。 詳しい利用方法は、HPCI共用ストレージ利用マニュアル をご覧ください。

マウントポイント
/gfarm/課題ID/利用者番号

利用状況の確認

利用者ポータル にログインし、 上部メニューの統計情報から、左メニューのHPCI統計をクリックすると利用状況が確認できます。

コア経過時間(秒)の値を3600で割って単位を「時間」にし、112(システムAのノード当たりコア数) で割っていただければ、 利用可能枠のノード時間に対する利用実績を算出することができます。 なお、このページではキューを利用可能な全ユーザの情報が表示されますので、それらを合計した値と課題割当時間を比較していただく必要があります。

情報共有CMS

HPCIの利用者に対して、各システム構成機関からの利用者向け情報発信ならびに、課題参加者が課題内でドキュメントを共有するための管理システムを運用しておりますのでご活用ください。

- 情報共有CMSのご案内

FX700の利用方法

概要

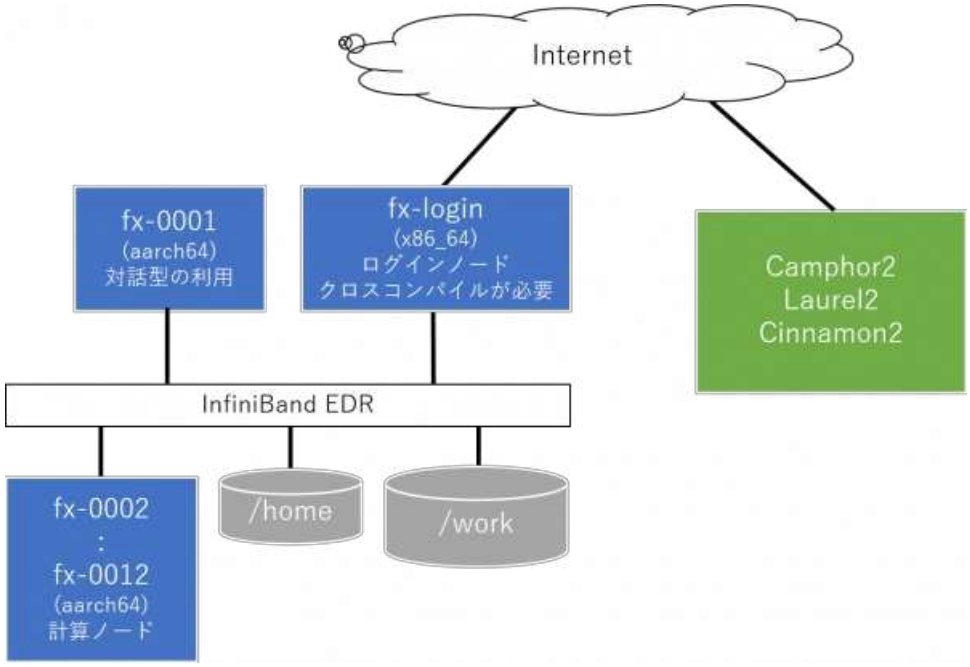
PRIMEHPC FX700は、富岳と同じCPUを搭載した最新のスーパーコンピュータです。京都大学学術情報メディアセンターでは、FX700を小規模な構成で保有しており、スーパーコンピュータシステムの利用者向けの評価環境として提供しています。

利用するためには、FX700「試用制度」及び「小ノード実行枠」に応募してください。試用制度であれば審査なし1か月間お使いいただけます。利用にかかる費用負担はありません。

2023年4月より、ジョブスケジューラを PBS から Slurm に変更しました。

システム構成

Camphor3やLaurel3とは独立したネットワークに接続した小規模なPCクラスタ型の構成です。 ログインノードは、Intel Xeonプロセッサ（x86_64アーキテクチャ）のサーバのため、計算ノードのA64FXプロセッサ（aarch64アーキテクチャ）でプログラムを実行するためには、クロスコンパイルが必要となります。利便性を考えて、計算ノード fx-0001 でも対話的にネイティブコンパイルが可能な構成としています。



計算ノードCPU仕様

項目	内容
名称	A64FX
命令	セット

項目	内容
アーキテクチャー	Armv8.2-A SVE
演算コア数	48 コア
クロック	1.8 GHz
理論演算性能	2.7648 TFLOPS

計算ノード仕様

項目	内容
アーキテクチャー	1 CPU/ノード
メモリ容量	32 GiB（HBM2, 4スタック）
メモリバンド幅	1,024 GB/s
インターコネクト	InfiniBand EDR
内蔵ストレージ	M.2 SSD Type 2280 スロット（NVMe）
OS	Red Hat Enterprise Linux 8

PRIMEHPC FX700の製品情報

ログインノード

FX700にログインするには、SSH公開鍵認証により次のホストに接続してください。

```
fx-login.kudpc.kyoto-u.ac.jp
```

- スパコンの利用者ポータルに登録済みの鍵でログインできます。（Camphor3, Laurel3のホームディレクトリにあるauthorized_keysに直接追加した鍵は利用できません）
- クラスタ内部はホストベース認証で相互ログイン可能です。

共有ストレージ

/home および /work を共有ストレージとして構成しています。/homeは fx-loginのローカルストレージですので、ログインノード上の操作に対する応答は高速ですが、負荷には強くありません。計算の実行には /work をご利用ください。

```
 [root@fx-0001 ~]# df -h /home/ /work/
Filesystem         Size  Used Avail Use% Mounted on
fx-login-ib:/home  399G   27G  373G   7% /home
armst-ib:/work     7.0T   25G  7.0T   1% /work
```

- /home は fx-login:/home をマウントしています。 ユーザ当たり**20GB** の容量制限をかけています。
- /work は armst:/work をマウント。計算にはこちらを使ってください。 ユーザ当たり **500GB** の容量制限をかけてい

ます。

- /home, /workはautofsによりアクセス時に都度マウントする仕様です。
- /work の容量が不足する場合はご相談ください。
- ファイルサーバの応答が無い場合は管理者にお知らせください。ファイルサーバ側を再起動します。
- 利用終了後にファイルは削除しますので、利用期間中にご自身でバックアップを実施してください。

コンパイル方法

利用可能なコンパイラは Fujitsu compiler と GCC です.

Fujitsu コンパイラ

fx-login におけるクロスコンパイル用コマンド

```
 # fortran用
[b12345@fx-login ~]$ frtpx -v
frtpx: Fujitsu Fortran Compiler 4.4.0
使用法: frtpx [オプション] ファイル

[b12345@fx-login ~]$ mpifrtpx -v
frtpx: Fujitsu Fortran Compiler 4.4.0
使用法: frtpx [オプション] ファイル

# C言語用
[b12345@fx-login ~]$ fccpx -v
fccpx: Fujitsu C/C++ Compiler 4.4.0
simulating gcc version 6.1

[b12345@fx-login ~]$ mpifccpx -v
fccpx: Fujitsu C/C++ Compiler 4.4.0
simulating gcc version 6.1
使用法: fccpx [オプション] ファイル

# C++言語用
[b12345@fx-login ~]$ FCCpx  -v
FCCpx: Fujitsu C/C++ Compiler 4.4.0
simulating gcc version 6.1
使用法: FCCpx [オプション] ファイル

[b12345@fx-login ~]$ mpiFCCpx -v
FCCpx: Fujitsu C/C++ Compiler 4.4.0
simulating gcc version 6.1
使用法: FCCpx [オプション] ファイル
```

fx-0001 におけるネイティブコンパイル用コマンド

```
 # fortran用
[b12345@fx-0001 ~]$ frt -v
frt: Fujitsu Fortran Compiler 4.4.0
使用法: frt [オプション] ファイル

[b12345@fx-0001 ~]$ mpifrt -v
frt: Fujitsu Fortran Compiler 4.4.0
使用法: frt [オプション] ファイル

# C言語用
[b12345@fx-0001 ~]$ fcc -v
fcc: Fujitsu C/C++ Compiler 4.4.0
simulating gcc version 6.1
使用法: fcc [オプション] ファイル

[b12345@fx-0001 ~]$ mpifcc -v
fcc: Fujitsu C/C++ Compiler 4.4.0
simulating gcc version 6.1
使用法: fcc [オプション] ファイル

# C++言語用
[b12345@fx-0001 ~]$ FCC -v
FCC: Fujitsu C/C++ Compiler 4.4.0
simulating gcc version 6.1
使用法: FCC [オプション] ファイル

[b12345@fx-0001 ~]$ mpiFCC -v
FCC: Fujitsu C/C++ Compiler 4.4.0
simulating gcc version 6.1
使用法: FCC [オプション] ファイル
```

代表的なコンパイルオプション

オプション名	説明
-Kopenmp	OpenMP指示子を有効にしてコンパイルします。
-Kparallel	自動並列化を有効にします。
-Kfast	ターゲットマシン上で高速に実行するオブジェクトプログラムを作成します。-O3 -Keval,fast_matmul,fp_contract,fp_relaxed,fz,ilfunc,mfunc,omitfp,simd_packed_promotion と等価です。
-KA64FX	A64FXプロセッサ向けのオブジェクトファイルを出力するように指示します。
-KSVE	Armv8-Aアーキテクチャの拡張であるSVEを利用するか否かを指示します。

コンパイラの詳細は、マニュアルを参照してください。

富士通コンパイラマニュアル（アクセスにはスパコンのログインアカウントで認証が必要）

GCC

fx-0001上でネイティブコンパイルしてください。

```
 [b12345@fx-0001 ~]$ gfortran --version
GNU Fortran (GCC) 8.3.1 20191121 (Red Hat 8.3.1-5)
Copyright (C) 2018 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

[b12345@fx-0001 ~]$ gcc --version
gcc (GCC) 8.3.1 20191121 (Red Hat 8.3.1-5)
Copyright (C) 2018 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

[b12345@fx-0001 ~]$ g++ --version
g++ (GCC) 8.3.1 20191121 (Red Hat 8.3.1-5)
Copyright (C) 2018 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

ジョブ投入方法 (Slurm)

sbatchコマンドでジョブを投入してください。Camphor3, Laurel3 とはジョブスクリプトの記述方法が異なりますのでご注意ください。 Camphor3, Laurel3 はSlurm を独自仕様にカスタマイズしていますが、FX700は標準のSlurmの機能のみを使用します。

バッチキュー

1ノードだけ割り当てたdebugキューと、10ノード割り当てた shortキューがあります。 debugキューでテストした後に、shortキューで本番のジョブを実行してください。

```
 $ sinfo -s
PARTITION AVAIL  TIMELIMIT   NODES(A/I/O/T) NODELIST
debug*       up   infinite         2/7/2/11 fx-[0002-0012]
short        up   infinite         2/7/2/11 fx-[0002-0012]
```

オプション

オプション	説明	初期値	最大値
#SBATCH -p *QUEUE*	キュー名の指定	--	--
#SBATCH -N *NODE*	使用するノード数の指定	1	試用制度:1 / 小ノード実行枠:8
#SBATCH -n *PROCS*	プロセス数の指定	1	試用制度:48 / 小ノード実行枠:384
#SBATCH --cpus-per-task=*CORES*	プロセスあたりのコア数の指定	1	48
#SBATCH --mem=*MEMORY*	プロセスあたりのメモリサイズの指定	650M	31200M

オプション	説明	初期値	最大値
#SBATCH -t *HOUR:MINUTES:SECONDS*	実行時間の上限を指定	1:00:00(1時間)	168:0:0(7日間)

ジョブスクリプトの記述例

ジョブスクリプト例1 (1コア8GBを確保した逐次ジョブ)

```
#!/bin/bash
#SBATCH -p debug
#SBATCH -N 1              # ノード数
#SBATCH -n 8              # プロセス数
#SBATCH --cpus-per-task=1 # プロセスあたりのコア数
#SBATCH --mem=8G          # ノード当たりのメモリサイズ

srun ./a.out
```

ジョブスクリプト例2 (16コア8GBを確保したOpenMPジョブ(16スレッド))

```
#!/bin/bash
#SBATCH -p debug
#SBATCH -N 1               # ノード数
#SBATCH -n 1               # プロセス数
#SBATCH --cpus-per-task=16 # プロセスあたりのコア数
#SBATCH --mem=8G           # ノード当たりのメモリサイズ
#SBATCH -t 1:00:00 # 経過時間の上限1時間

srun ./a.out
```

ジョブスクリプト例3 ((48コア、8GBメモリ) x 2ノードのMPIジョブ(96プロセス))

```
#!/usr/bin/bash
#SBATCH -p debug
#SBATCH -N 2               # ノード数
#SBATCH -n 96              # プロセス数
#SBATCH --cpus-per-task=1  # プロセスあたりのコア数
#SBATCH --mem=30G          # ノード当たりのメモリサイズ
#SBATCH -t 1:00:00 # 経過時間の上限1時間

srun ./a.out
```

ジョブスクリプト例4 ((48コア、30GBメモリ) x 2ノードのハイブリッドジョブ(24プロセス x 4スレッド))

```
 #!/bin/bash
#SBATCH -p debug
#SBATCH -N 2               # ノード数
#SBATCH -n 24              # プロセス数
#SBATCH --cpus-per-task=4  # プロセスあたりのコア数
#SBATCH --mem=30G          # ノード当たりのメモリサイズ
#SBATCH -t 1:00:00 # 経過時間の上限1時間

srun ./a.out
```

ネットワークストレージサービス

概要

2022年5月より、研究データの保存環境ならびにデータ収集基盤の拡充を目的として、ネットワークストレージサービスの試行を開始しました。 利用者ポータル から利用開始処理を行っていただくことでご利用いただけます。 試行期間中は、予告なくメンテナンスの実施ならびにサービス内容の見直しを行うことがありますので、予めご了承ください。

活用例

- 学認RDMとのストレージ連携による研究データ共有
- IoT機器などのデータ収集
- スパコンストレージのバックアップ

システム構成

スーパーコンピュータシステムとは独立したネットワークに接続した小規模なネットワークストレージクラスタです。

項目	内容
機器	DELL EMC isilon A200 x 4台 DELL EMC PowerScale F200 x 4台
ストレージ容量	260TB
フロントエンドネットワーク	5Gbps

サービス提供内容

大型計算機システム利用者番号 (アカウント)をお持ちの方は、以下のサービスを無償でご利用いただけます。 **本サービスは、システム管理者による利用者のデータバックアップは行いません。**利用される際は、登録したデータの消失等に備え、バックアップ等の対策を必要に応じて実施してください。

項目	内容
対応プロトコル	FTPS ならびに S3 (Amazon S3互換、パス形式)
ストレージ容量	100GB
転送量	無制限
バックアップ	なし
FTPアカウント	1つ
S3アカウント	1つ

項目	内容
S3バケット数	1つ
S3リクエスト数	無制限
S3オブジェクト数	無制限
機器のタイムゾーン	UTC

利用開始処理

1. 利用者ポータル にアクセスし、大型計算機システム利用者番号でログインしてください。
2. 左カラムの「ネットワークストレージサービス」を押下してください。

3. FTPで使用するパスワードを発行する場合は赤色、S3のシークレットキーを発行する場合は青色で囲まれたドロップダウンリストから、「生成(再生成)する」を選択し、同意事項に同意いただいた上で、「送信」ボタンを押下してください。

4. 正常にパスワード/シークレットキーが発行されましたら、画面上部にパスワード情報が表示されます。パスワード情報は **1**度しか表示されません ので、ご注意ください。

接続方法(S3)

接続情報

項目	内容
接続先	s3.kudpc.kyoto-u.ac.jp:443
アクセスキー	利用者ポータルをご確認ください
シークレットキー	利用者ポータルをご確認ください
バケット名	利用者ポータルをご確認ください

awscliを使用する場合

インストール方法

公式マニュアルをご覧ください。 なお、スーパーコンピュータシステムではモジュールファイルを用意しています。詳細はこちらをご覧ください。

初期設定

```
$ aws configure --profile kudpc
AWS Access Key ID [None]:     ## 利用者ポータルで発行したアクセスキーを入力
AWS Secret Access Key [None]: ## 利用者ポータルで発行したシークレットアクセスキーを入力
Default region name [None]:   ## 入力せずEnter
Default output format [None]: ## 入力せずEnter
```

ファイルの操作方法

```
# 基本的なコマンド体系
$ aws --profile kudpc --endpoint-url=https://s3.kudpc.kyoto-u.ac.jp s3 [Command]

## ファイルの一覧を取得
$ aws --profile kudpc --endpoint-url=https://s3.kudpc.kyoto-u.ac.jp s3 ls s3://b59999

## ファイルをアップロード
$ aws --profile kudpc --endpoint-url=https://s3.kudpc.kyoto-u.ac.jp s3 cp sample.sh s3://b59999

## ファイルをダウンロード
$ aws --profile kudpc --endpoint-url=https://s3.kudpc.kyoto-u.ac.jp s3 cp s3://b59999/sample.sh ./

## ファイルをネットワークストレージから削除
$ aws --profile kudpc --endpoint-url=https://s3.kudpc.kyoto-u.ac.jp s3 rm s3://b59999/sample.sh

## ローカルのフォルダをネットワークストレージに同期させる場合
$ aws --profile kudpc --endpoint-url=https://s3.kudpc.kyoto-u.ac.jp s3 sync ./ s3://b59999
```

s3fsを使用する場合

インストール方法

公式マニュアルをご確認ください。 なお、スーパーコンピュータシステムではモジュールをロードすることなくご利用いただけます。

初期設定

```
## ネットワークストレージをマウントするディレクトリを作成
$ mkdir ./kudpc-s3

## S3の認証情報を作成
$ echo "{アクセスキー}:{シークレットキー}" > ${HOME}/.passwd-s3fs
$ chmod 600 ${HOME}/.passwd-s3fs
```

接続方法

```
## マウント
$ s3fs b59999 ./kudpc-s3 -o use_path_request_style,url=https://s3.kudpc.kyoto-u.ac.jp,uid=$(id -u),gid=
## アンマウント
$ fusermount -u ./kudpc-s3
```

Cyberduckを使用する場合

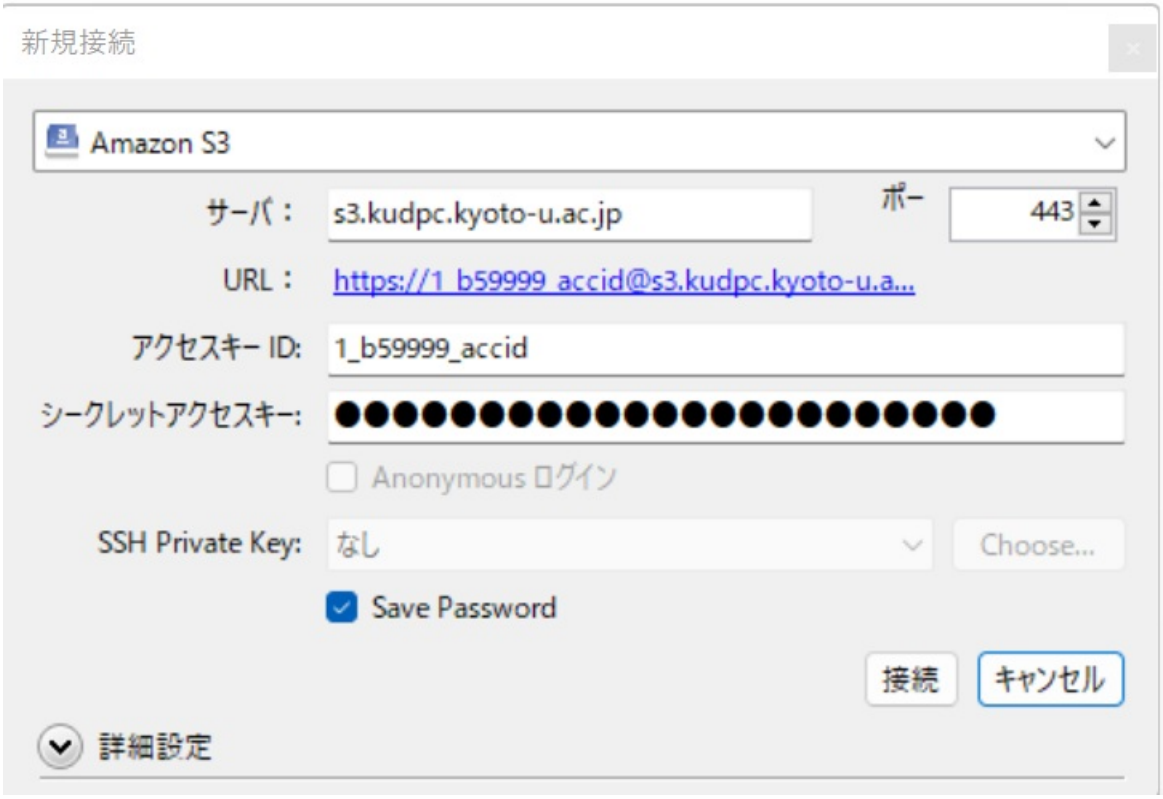
インストール方法

公式サイトよりダウンロードしてご利用下さい。

接続方法

1. Cyberduckを起動してください。
2. 画面上部の「新規接続」を押下してください。
3. 以下の表の通りセッション情報を入力し、「接続」ボタンを押下してください。

項目	設定内容
プロトコル選択画面	Amazon S3
サーバ	s3.kudpc.kyoto-u.ac.jp
ポート	443
アクセスキーID	S3のアクセスキー
シークレットアクセスキー	S3のシークレットキー



接続方法(FTP)

接続情報

項目	内容
接続先	s3.kudpc.kyoto-u.ac.jp
アカウント名	利用者ポータルをご確認ください
パスワード	利用者ポータルをご確認ください

WinSCPを使用する場合

インストール方法

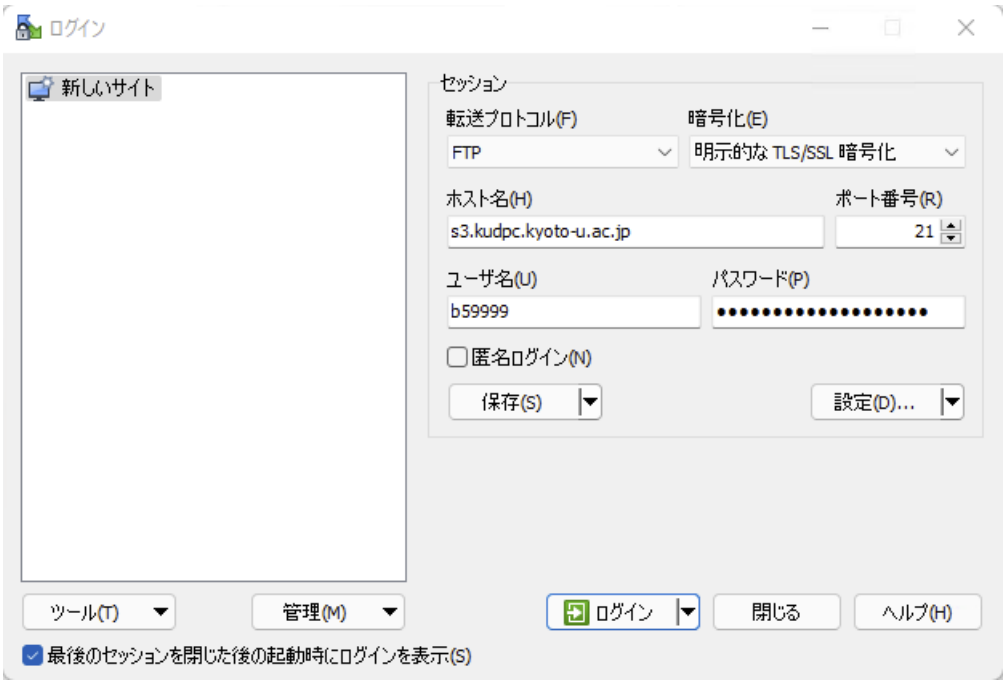
WinSCPのインストールをご覧ください。

接続方法

1. WinSCPを起動してください。
2. 以下の表の通りセッション情報を入力し、「ログイン」ボタンを押下してください。

項目	設定内容
転送プロトコル	FTP
暗号化	明示的なTLS/SSL 暗号化
ホスト名	s3.kudpc.kyoto-u.ac.jp

項目	設定内容
ポート番号	21
ユーザ名	FTPのアカウント名
パスワード	FTPのパスワード



3. 成功するとWinSCPの画面が表示され、ファイル転送が可能になります。

Cyberduckを使用する場合

インストール方法

公式サイトよりダウンロードしてご利用下さい。

接続方法

1. Cyberduckを起動してください。
2. 画面上部の「新規接続」を押下してください。
3. 以下の表の通りセッション情報を入力し、「接続」ボタンを押下してください

項目	設定内容
プロトコル選択画面	FTP(ファイル転送プロトコル)
サーバ	s3.kudpc.kyoto-u.ac.jp
ポート	21
ユーザ名	FTPのアカウント名
パスワード	FTPのパスワード

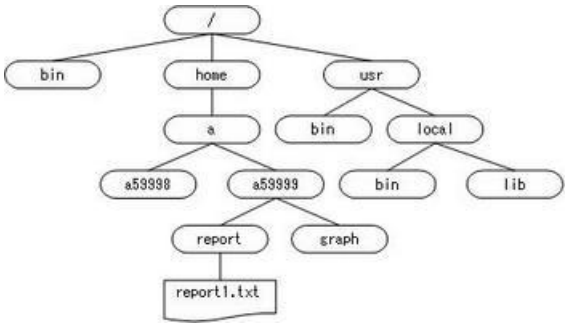

FTP（ファイル転送プロトコル）
サーバ：
s3.kudpc.kyoto-u.ac.jp
ポート：
21
URL：
ftp://s3.kudpc.kyoto-u.ac.jp
ユーザ名:
b59999
パスワード:
Anonymous ログイン
SSH Private Key:
なし
キーチェーンに追加する
キャンセル
接続

UNIX/Linuxの基礎知識

ファイルシステム

ファイルとディレクトリ

ユーザーが作成した様々なデータは、 ファイル という形で記録されます。そして、記録されたファイルを ディレクトリ と呼ばれる入れ物に入れて管理します。ディレクトリの中にディレクトリを作って、階層的にファイルを管理することも可能です。



このようなファイル構造を、木の枝になぞらえてツリー構造と呼んでいます。 UNIX では、木の根っこに当たる部分を ルートディレクトリ と呼び、“/（スラッシュ）”を使って表します。そして、ユーザーが作業を行なっているディレクトリを カレントディレクトリ と呼びます。

絶対パスと相対パス

ツリー構造においては、ファイルを指定するための パス という概念が必要となります。パスの指定方法には、 絶対パス と 相対パス の２種類があります。

- 絶対パス

ルートディレクトリを基準にしてファイルを指定する方法です。

（例）図1「report1.txt」を指定する場合。

```
/home/a/b59999/report/report1.txt
```

- 相対パス

カレントディレクトリを基準にしてファイルを指定する方法です。

（例）カレントディレクトリ「/home/a/b59999」を基準にして、図1「report1.txt」 を指定する場合。

```
report/report1.txt
```

基本的なコマンド

UNIXでよく使われる基本的なコマンドを説明します。

pwd ーカレントディレクトリを表示する

```
pwd
```

実行例

```
[b59999@hx001 ~]$ pwd
```

```
/home/a/b59999          # カレントディレクトリが表示されている
```

ls ーファイルの一覧を表示する

```
ls [オプション] [ファイル名ディレクトリ名]
```

主なオプション

オプション	効果
-l	ファイルの詳しい情報を表示する
-F	ディレクトリには“/（スラッシュ）”、実行可能ファイルには“*（アスタリスク）”等をつけて属性をわかりやすく表示する
-a	システム等で使用される、“.（ドット）”で始まるファイルも表示する

実行例

```
[b59999@hx001 ~]$ ls
```

```
file1.txt file2.bmp dir1          # ファイルが表示されている
[b59999@hx001 ~]$ ls -a
```

```
. .cshrc .tcshrc file1.txt dir1   # '.' で始まるファイルも表示されている
.. .login .bashrc file2.bmp
[b59999@hx001 ~]$ ls -F
```

```
file1.txt file2.bmp dir1/         # ディレクトリには '/' がついている
[b59999@hx001 ~]$
```

cd ーカレントディレクトリを移動する

```
cd [ディレクトリ名]
```

実行例

```
[b59999@hx001 ~]$ pwd
```

```
/home/a/b59999              # カレントディレクトリは /home/a/b59999
```

```
[b59999@hx001 ~]$ cd dir1  # カレントディレクトリを dir1 に移動する
[b59999@hx001 ~]$ pwd
```

```
/home/a/b59999/dir1         # カレントディレクトリが dir1 に移動した
```

cp ーファイルやディレクトリをコピーする

```
cp [オプション] [コピー元] [コピー先]
```

主なオプション

オプション	効果
-R	コピー元がディレクトリだった場合、そのディレクトリ以下のツリー構造を含めてコピーします。

実行例

```
[b59999@hx001 ~]$ ls -F
file1.txt dir1/
```

```
[b59999@hx001 ~]$ cp file1.txt file2.txt  # file1.txtをfile2.txtとしてコピーする
[b59999@hx001 ~]$ ls -F
```

```
file1.txt file2.bmp dir1/              # file2.txt ができている
```

```
[b59999@hx001 ~]$ cp file2.txt dir1    # file2.txt を dir1 にコピーする
[b59999@hx001 ~]$ ls dir1
```

```
file2.txt         # file2.txt が dir1 にコピーされている
```

```
[b59999@hx001 ~]$ cp -R dir1 dir2      # dir1 を dir2 にコピーする
[b59999@hx001 ~]$ ls -F
```

```
file1.txt file2.txt dir1/ dir2/        # dir2 ができている
[b59999@hx001 ~]$ ls dir2
```

```
file2.txt         # ツリー構造も含めて dir1 が dir2 にコピーされている
[b59999@hx001 ~]$
```

mkdir ーディレクトリを新規作成する

```
mkdir [ディレクトリ名]
```

実行例

```
[b59999@hx001 ~]$ ls
file1.txt file2.bmp
[b59999@hx001 ~]$ mkdir dir1
[b59999@hx001 ~]$ ls -F
```

```
file1.txt file2.bmp dir1/          # dir1 ができている
[b59999@hx001 ~]$
```

rm ーファイルやディレクトリを削除する

```
rm [オプション] [ファイル名ディレクトリ名]
```

主なオプション

オプション	効果
-R	対象がディレクトリだった場合、そのディレクトリ以下のツリー構造を含めて削除する

実行例

```
[b59999@hx001 ~]$ ls -F
file1.txt file2.bmp dir1/
```

```
[b59999@hx001 ~]$ rm file1.txt    # file1.txt を削除する
[b59999@hx001 ~]$ ls -F
```

```
file2.bmp dir1/                   # file1.txt が削除されている
[b59999@hx001 ~]$ ls -F dir1
```

```
dir2/          # dir1 の下に dir2 がある
```

```
[b59999@hx001 ~]$ rm -R dir1      # dir1 を削除する
[b59999@hx001 ~]$ ls -F
```

```
file2.bmp          # dir1 がそれ以下のツリー構造ごと削除されている
```

mv ーファイルを移動する(ファイル名を変更する)

`mv` コマンドは、ファイルやディレクトリを移動させる時に使用します。また、同じディレクトリに移動させることで、ファイル名を変更することもできます。

```
mv［元のファイル名］［新しいファイル名］
```

実行例

```
[b59999@hx001 ~]$ ls -F
file1.txt file2.bmp dir1/
```

```
[b59999@hx001 ~]$ mv file1.txt dir1         # file1.txt を dir1 に移動する
[b59999@hx001 ~]$ ls -F
```

```
file2.bmp dir1/          # file1.txt が無くなっている
[b59999@hx001 ~]$ ls dir1
```

```
file1.txt          # dir1 に移動している
```

```
[b59999@hx001 ~]$ mv file2.bmp file3.bmp    # file2.bmp を file3.bmp に移動する
[b59999@hx001 ~]$ ls -F
```

```
file3.bmp dir1/          # file3.bmp にファイル名が変わっている
```

cat ーファイルの中身を画面に表示する

```
cat［ファイル名］
```

実行例

```
[b59999@hx001 ~]$ ls -F
file1.txt file2.bmp dir1/
[b59999@hx001 ~]$ cat file1.txt
```

```
abcdefghijklmnopqrstuvwxyz          # file1.txt の中身が表示されている
[b59999@hx001 ~]$
```

more と less ーファイルの中身を表示する

`cat` コマンドに代わって `more` コマンド、 `less` コマンドを使用すると、ファイルが長い場合1ページごとに表示を止め、コマンド待ち状態になります。

```
more（もしくは less ） [ファイル名]
```

実行例

```
[b59999@hx001 ~]$ more file3.c
#include <stdio.h>    /* file3.c の中身が表示されている */
#include <math.h>
#include <stdlib.h>
 (・・中略・・)
    for(i=0; i<10; i++){
        a[i] = b[i]*c[i];
--More--(10%)      ← 1 ページ表示したところでコマンド待ち
                     スペースを押すことで次の 1 ページに進む
```

logout ーログアウトする

```
logout
```

実行例

```
[b59999@hx001 ~]$ logout          # ログアウトする
```

man -マニュアルを表示する(各コマンドの説明を見る)

lsコマンドの説明を表示する

```
[b59999@hx001 ~]$ man ls
```

`man` コマンドのメッセージを日本語表示する場合は、sshクライアントの文字コードの設定をUTF-8に指定した上で、環境変数LANGにja_JP.UTF-8を設定してください。

tcshを使用している場合

```
[b59999@hx001 ~]$ setenv LANG ja_JP.UTF-8
```

bashを使用している場合

```
[b59999@hx001 ~]$ export LANG=ja_JP.UTF-8
```

vi エディタの使い方

vi エディタは、UNIX 系 OS に標準で組み込まれているテキストエディタです。操作体系が非常に特殊なので、以下に詳しく説明します。

vi エディタの起動

```
vi [ファイル名]
```

2種類のエディットモード

vi エディタのエディットモードは、 コマンドモード と エディットモード の2種類に大きく分けられます。

- コマンドモード

検索や置換、ファイルの保存、文字や行の削除など、文字入力以外の作業を行ないます。

- エディットモード

実際に文字を入力します。

エディタを起動すると、まずコマンドモードが立ち上がります。 `i` 、 `I` 、 `a` 、 `A` 、 `o` 、 `O` などの挿入コマンドでエディットモードに入ることができます。コマンドモードに復帰する時は、ESCキーを押してください。

---

vi コマンドリファレンス

カーソルの移動

文字指向のジャンプ	
h , j , k , l	左、下、上、右 （ ← , ↓ , ↑ , → ）

テキスト指向のジャンプ	
w , W , b , B	前/後の単語
e , E	単語の末尾
) , (	次/前の文の先頭
} , {	次/前の段落の先頭
]] , [[	次/前のセクションの先頭

行指向のジャンプ	
0(ゼロ) , $	カレント行の先頭/末尾
^	カレント行の先頭（空白以外の）文字
+ , ?	次/前の行の先頭の文字
_n_l	カレント行の *n* 文字目
H	画面の最上行
M	画面の中央行
L	画面の最下行
_n_H	上から *n* 行目の行
_n_L	下から *n* 行目の行

画面指向のジャンプ	
CTRL + f , CTRL + b	次/前の画面にスクロール
CTRL + d , CTRL + u	上/下に半画面分スクロール
CTRL + e , CTRL + y	ウィンドウの下/上にもう1行表示
z RETURN	カーソルのある行を画面の一番上に表示
z. （ゼット　ドット）	カーソルのある行を画面中央に表示
z- （ゼット　ハイフン）	カーソルのある行を画面の一番下に表示
CTRL + l , CTRL + r	画面の書き直し（スクロールなし）

検索	
/ *pattern*	パターンを順方向に検索
? *pattern*	パターンを逆方向に検索

検索	
n , N	最後の検索を、同じ/反対の方向で繰り返す
/ , ?	直前の検索を、順方向/逆方向に繰り返す
f*x*	カレント行内の、カーソル位置より後にある *x* にジャンプ
F*x*	カレント行内の、カーソル位置より前にある *x* にジャンプ
t*x*	カレント行内の、カーソル位置より後にある *x* の直前の文字にジャンプ
T*x*	カレント行内の、カーソル位置より前にある *x* の直後の文字にジャンプ
;	直前のカレント行内検索を繰り返す
,	直前のカレント行内検索を反対方向で繰り返す

行番号によるジャンプ	
CTRL + g	カレント行の行番号を表示
_n_G	行番号 *n* にジャンプ
G	ファイルの最終行にジャンプ
:*n*	行番号 *n* にジャンプ

位置のマーク	
m*x*	現在位置を *x* としてマーク
‘*x*	*x* にジャンプ
“	直前のマークまたは文脈に戻る
`*x*	マークを *x* を含む行の先頭にジャンプ
``	直前のマークを含む行の先頭に戻る

終了コマンド

終了コマンド	
ZZ	ファイルを書き込んで（保存）終了
:x	ファイルを書き込んで（保存）終了
:wq	ファイルを書き込んで（保存）終了
:w	ファイルの書き込み（保存）
:w!	（強制的な）ファイルの書き込み（保存）
:q	ファイルの編集を終了
:q!	ファイルの編集を終了（全ての変更を無効にする）

終了コマンド	
:e!	現在のファイルを、最後に書き込んだ（保存した）時点の内容に戻す

編集コマンド

挿入	
`i`，`a`	カーソルの前/後にテキストを挿入
`I`，`A`	行の先頭/末尾にテキストを挿入
`o`，`O`　（大文字　オー）	カーソルの下/上にテキスト入力用の新しい行をオープン
変更	
`r`	文字を変更
`cw`	単語を変更
`cc`	カレント行を変更
`C`	行末まで変更
`R`	文字列を重ね書きする
`s`	文字をテキストで置き換える
`S`	カレント行をテキストで置き換える
移動、削除	
`x`	文字の削除
`X`	カーソルの前にある文字の削除
`dw`	単語の削除
`dd`	カレント行の削除
`D`	行末まで削除
`p`，`P`	削除したテキストをカーソルの後/前に挿入する
“_n_p	カーソルの後に、削除バッファ番号 *n* のテキストを挿入する（最新 9 回の削除について有効）
ヤンク	
`yw`	単語のヤンク（コピー）
`yy`	カレント行のヤンク
“_a_yy	*a* という名前のバッファにカレント行をヤンク
`p`，`P`	ヤンクしたテキストをカーソルの後/前に挿入する
“_a_p	カーソルの前に、バッファ *a* のテキストを挿入する

その他のコマンド	
.（ドット）	最後の編集コマンドの繰り返し
`u`，`U`	最後の編集の取り消し（ undo ）/カレント行の復元
`j`	2 行の連結

`ex` コマンドを使った場合	
:d	行の削除
:m	行の移動
:co , :t	行のコピー
/ , ?	直前の検索を、順方向/逆方向に繰り返す
:.,$d	カレント行からファイルの最終行まで削除

Emacs の使い方

Emacs とは

Emacs は、UNIX 系の OS でよく利用される、高機能でカスタマイズ性の高いテキストエディタです。
Emacs で編集を行う際は、バッファにファイルを読み込んで作業した後、編集したバッファをファイルに書き込む形になります。複数のバッファを立ち上げ、切り替えながら作業できるので、バッファごとに名前がついています。通常は、編集するファイル名がバッファの名前となっています。

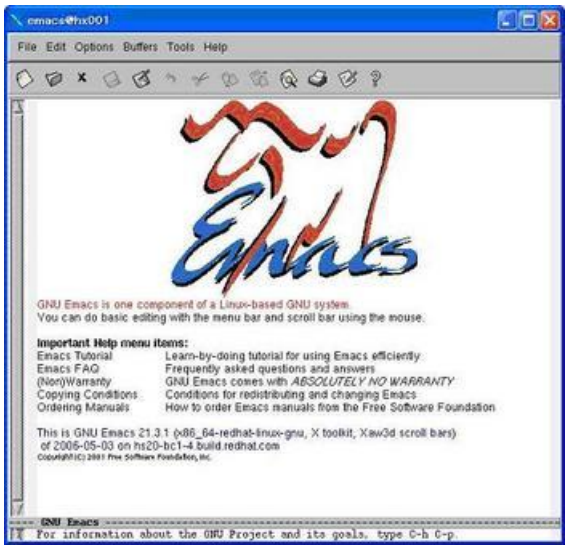
Emacs の起動

```
emacs［ファイル名］
```

起動時にファイル名を指定すると、その名前で編集バッファが用意され、該当ファイルが読み込まれます。（ファイルが存在しない場合は、バッファは空っぽのままです。）
ファイル名を省略して起動すると、バッファ名は自動的に **scratch** となります。

Emacs の画面

X（ウィンドウシステム）が利用できる環境で起動すると、Xクライアントとして新規ウィンドウが立ち上がります。



ウィンドウシステムが利用できない環境の場合は、次のような画面が表示されます。

Emacs の画面は、大きく3つの部分に分かれています。

- テキストウィンドウ

一番大きいエリア。入力した文字が表示されます。

- モード行

テキストウィンドウの下にある反転した行。テキストウィンドウの状態などが表示されます。
モード行のおおよその意味は、次のとおりです。

- エコー行

モード行の下にある行。 **Emacs** からのメッセージが表示されます。入力もここで行います。

Emacs の基本操作

カーソル移動などの操作は、特殊なキー入力を使います。 キー入力の方法は次の二つです。

- **CTRL** キー を押しながら文字キーを押す（ **C-**文字キー ）
- **ESC** キー を押した後に文字キーを押す（ **M-**文字キー ）

ファイルを開く（ `C-x` `C-f` ）

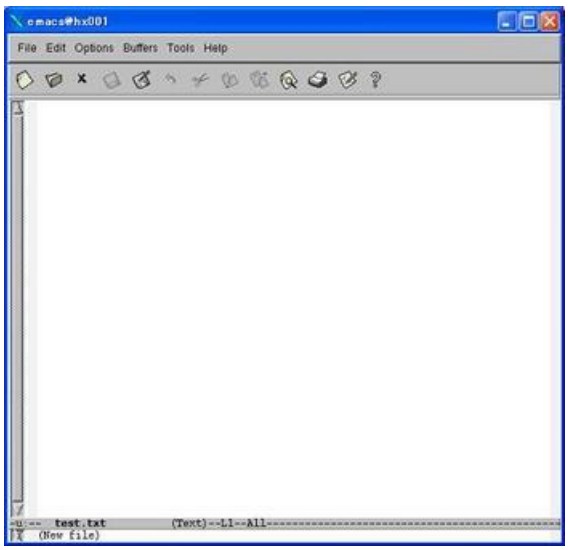
Emacs 起動後、新たにファイルを読み込みたい時は、 `C-x` `C-f` （ CTRL を押しながら x 、続いて CTRL を押しながら f ）を押します。すると、エコー行に次のメッセージが表示されます。

```nohighlight
Find file: ~/
```

Emacs をホームディレクトリ以外で起動した場合は、/~ ではなく、カレントディレクトリのパスが表示されます。
メッセージの後に続けて、開きたいファイル名（ 例　test.txt ）を入力し、Enter キーを押します。

```
Find file: ~/test.txt
```

すると、test.txt という名前のバッファが用意され、ファイルの内容が編集バッファに読み込まれます。ファイルが存在しない場合は、次のように新規ファイルを意味する（ New file ）のメッセージがエコー行に表示されます。



文字の入力

文字は、テキストウィンドウ上のカーソル位置に入力されます。文字の削除は、Delete キーです。（Backspace キーは文字の削除には使用しません。）

日本語入力システム

日本語を入力する場合は、まず `C-¥` （ CTRL を押しながら ¥ [バックスラッシュ] ）で日本語入力システムを起動させてください。システムが起動するとモード行の左隅の表示が [? -] から [あ] に切り替わり、ローマ字かな変換が可能となります。入力システムを元に戻す時は、ふたたび `C-¥` を押してください。

日本語入力システムで、ｋｙｏｕｔｏｄａｉｇａｋｕとタイプすると、次のように表示されます。

```
| きょうとだいがく |
```

左右に表示される縦棒“|”は、この縦棒に挟まれた区間に、かな漢字変換が適用されることを意味しています。変換はSpace キーで行います。この時、左隅の表示は[あ] から [漢] に切り替わります。

```
| 京都大学 |
```

続けて Space キーを押すと、次の変換候補が表示されます。Enter キーまたは `C-l` で入力を確定します。変換する文節を変えたい時は、 `C-i` または `C-o` を押してください。文節毎に変換を行なう場合は、文節にカーソルを移動してSpace キーを押します。漢字変換を取り消す際は `C-c` を押してください。

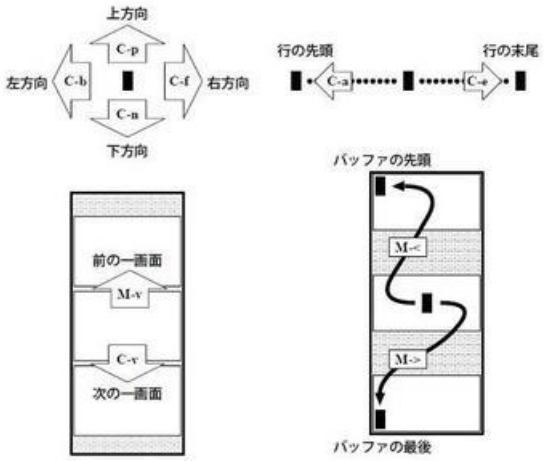
カーソルの移動

カーソル移動の方法は次のとおりです。

col 1	col 2
`C-p`	上方向に移動
`C-n`	下方向に移動
`C-f`	右に移動
`C-b`	左に移動
`C-a`	行の先頭に移動
`C-e`	行の末尾に移動

キーはそれぞれ previous、next、forward、backward、ahead、end of lineの略です。

col 1	col 2
`C-v`	次の画面にスクロール
`M-v` （ESC を押して離し v を押す）	前の画面にスクロール
`M-<`	バッファの先頭に移動
`M->`	バッファの最後に移動



削除とコピー

文字の削除には、いくつか方法があります。

col 1	col 2
Delete キー	一文字戻って削除
`C-d`	カーソル位置にある文字を一文字削除
`C-k`	カーソル位置から行の末尾までを全て削除

範囲を指定して削除することも可能です。まず指定したい範囲の先頭にカーソルを移動させ、 `C-` @ でマークを付けます。その際、エコー行に Mark set と表示されます。次に、範囲の最後にカーソルを移動させ `C-w` を押します。すると、現在のカーソル位置からマークを付けた範囲が全て削除されます。

`C-k` や `C-w` で削除した文字はコピーバッファに保存されるため、 `C-y` でペーストすることができます。コピーバッファは、 `C-k` や `C-w` を実行する毎に書き換えられます。
ただし、 `C-d` や Delete キーで削除した文字は、コピーバッファに保存されませんので注意してください。

操作の取り消し

操作を取り止める時は `C-g` を押します。この時、エコー行には Quit と表示されます。直前の編集操作を取り消す時は、 `C-x` `u` を押します。この時、エコー行には Undo! と表示されます。

ファイルの保存（ `C-x` - `C-s` ）

編集内容をファイルに保存する際は、 `C-x` `C-s` を押します。この時、 **scratch** バッファで編集作業を行なっていた場合は、エコー行に File to save in: ~/ と表示されますので、ファイル名を指定して保存してください。

編集バッファの抹消

編集バッファを抹消する際は、 `C-x` `k` を押します。
エコー行に Kill buffer: (default test.txt) と表示されますので、Enter キーを押してください。内容に変更がなければ、そのまま抹消されます。バッファの内容に変更があった場合は、さらに Buffer test.txt modified; kill anyway? (yes or no) のメッセージが表示されます。変更内容を無効にして抹消を実行する場合は yes を、抹消を取り止める場合は no を入力して Enter キーを押してください。

Emacs の終了

Emacs を終了する際は、 `C-x` `C-c` を押します。
保存していない編集バッファがある場合、エコー行に次のようなメッセージが表示されます。
（ファイル名は一例です。）

```
Save file /home/a/b59999/test.txt? (y, n, !, ., q, C-r or C-h)
```

ファイルを保存して終了する際は y を、保存せずに終了する場合は n を入力してEnterキーを押してください。
n の場合は、続いてエコー行に確認メッセージが表示されますので、本当に終了するなら yes 、終了を取り止めるなら no を入力してください。

```
Modified buffers exist; exit anyway? (yes or no)
```

Emacs をマスターするために

Emacs には、チュートリアル（個人指導）が用意されています。開始するには `M-?` `t` を押します。ここで紹介していない数多くの機能を学習できるので、 **Emacs** を早くマスターしたい方にはおすすめです。
チュートリアルもまた一つのバッファで実行されますので、終了の際は、 `C-x` `k` でバッファを抹消してください。

その他

本章では、システムの利用に関する各種情報を掲載します。

- [ポータル] 利用開始手続き
- 講習会アカウントの有効化

[ポータル] 利用開始手続き

スーパーコンピュータの利用者番号の発行が完了したら、Eメールにて登録完了通知を送付いたします。 登録完了通知を受け取ったら、以下の利用開始手続きを行ってください。

なお、前年度から継続してご利用中の方は、本作業は不要です。

手続きの流れ

初回の利用開始時に以下の手順で利用開始手続きを実施してください。 このページでは、以下の１～４の作業について具体的な画面と共に説明します。

支払責任者も同様な手順で利用開始が可能です。利用者番号を支払責任者番号に、利用承認書を登録確認通知書に置き換えてご確認をお願いします。

1. Eメールによる登録完了通知を受領し、利用者番号を確認する
2. 利用者ポータルより、初回の利用開始の手続きを要求する
3. 登録Eメールアドレスに届いたワンタイムURLを開き、利用承認書のダウンロード、パスワードの設定を実施する
4. 以降、パスワード認証により利用者ポータルにログイン可能となる (引き続き、スパコンの公開鍵認証の設定等を実施)

1. 登録完了通知の受領

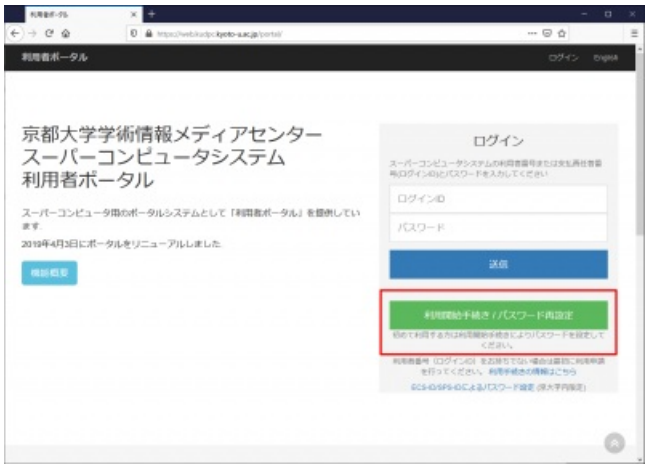
登録完了通知として、下記のメールを送付しております。 一部表現は簡略化していますが、実際は日英併記のメールを送付いたします。 このメールに記載の利用者番号を利用者ポータルやスーパーコンピュータのログインに使用します。忘れないようにしてください。

```
Subject: [京大スパコン] 継続登録完了のお知らせ (20xx 年度)
From: 京大スパコン利用者ポータル
-------------------------------------------------------------------------------------------
スーパーコンピュータのアカウントの登録が完了しました。
以下のURLより、利用承認書のダウンロード及びパスワード設定を
行うことで、利用開始することができます。

利用者番号： b59999

利用開始手続きURL：
https://web.kudpc.kyoto-u.ac.jp/portal/...略

スーパーコンピュータを利用して得られた研究成果を論文等で発表された
場合は、以下のURLに記載の方法で報告してください。
http://www.iimc.kyoto-u.ac.jp/ja/services/comp/apply/report.html

スーパーコンピュータシステムの使い方
https://web.kudpc.kyoto-u.ac.jp/manual/ja

----
京都大学学術情報メディアセンター
スーパーコンピュータシステム
```

2.利用開始手続きの要求

利用者ポータルへの接続

WEBブラウザを起動し、登録確認通知メールに記載のURLにアクセスしてください。 若しくは、利用者ポータル (https://web.kudpc.kyoto-u.ac.jp/portal/) のトップページより、以下の画像の赤四角で囲んだ利用開始手続きボタンを押下してください。



利用開始手続きの要求

以下の画面が表示されたら、登録完了通知に記載の利用者番号または支払責任者番号と、利用申請書に記載したEメールアドレスを入力してください。 利用開始手続きを実施するための、ワンタイムURLをEメールアドレスで送付いたします。

なお、本手続きはオンラインで実施するため、所属機関が発行したEメールアドレスでなければ実施できません。 登録したEメールアドレスに問題がある場合は、申請窓口に相談してください。

利用開始手続きのメール内容

以下のようにユニークな長い文字列を含むURLを通知いたします。 1時間だけ有効なURLですので、失効した場合は上記手順を再度実行してください。

Subject: [京大スパコン] 利用開始手続きについて
From: 京大スパコン利用者ポータル

京都大学学術情報メディアセンタースーパーコンピュータシステムの
利用開始手続きに関するお知らせです。

手続きを進めるには、以下のURLをクリックし手続きを進めてください。

必要な手続き
・利用承認書または登録確認通知書のダウンロード
・パスワード設定

[URL]
https://web.kudpc.kyoto-u.ac.jp/portal/...略.../c2cxwupu6jobyn...略
* URLの有効期限は1時間です。
* 接続時にエラーが出た場合は、再度申込みの手続きをお願いします。

--
京都大学学術情報メディアセンター
スーパーコンピュータシステム
利用者ポータル

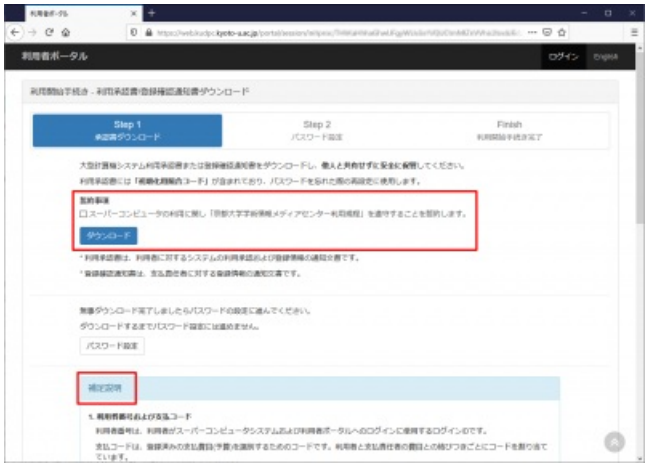
3. 利用開始手続き

上記2の作業で届いたEメールに記載のワンタイムURLにアクセスし、利用承認書または登録確認通知書をダウンロードしてください。 ダウンロード後にパスワードの設定を行うことができます。

承認書のダウンロード

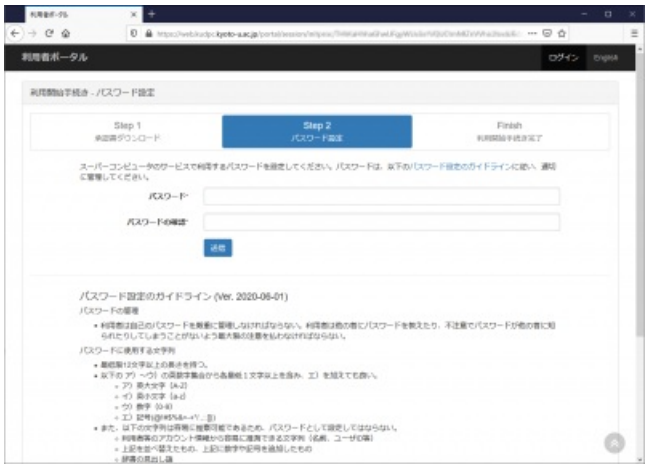
ワンタイムURLにアクセスすると、以下の画面が開きます。ダウンロードボタンを押下することで利用承認書（PDF）をダウンロードすることができます。 利用承認書には登録情報とパスワードを忘れた際に使用する照合コードを記載しています。他人に情報が漏れることが無いように安全に保管してください。 この画面ではダウンロードは1回しかできませんが、利用者ポータルにログインできるようになれば再度ダウンロードが可能です。

以下の画像では省略していますが、ページの下方に利用に関する補足説明を記載していますのでご一読ください。



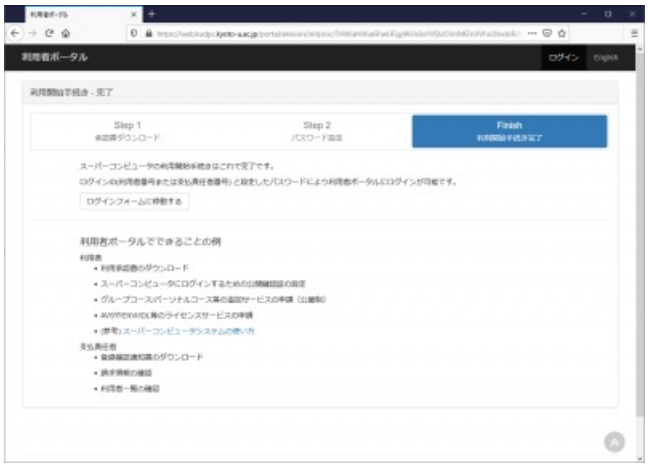
パスワードの設定

続いてパスワードを設定してください。ここで設定するパスワードはスーパーコンピュータ関係のサービスで使用するパスワードとなります。



手続きの完了

パスワード設定が完了すると以下の画面に遷移します。 以降は、利用者番号と登録したパスワードを用いて利用者ポータルにログインすることが可能です。 利用者ポータルでは、スーパーコンピュータで利用する公開鍵の登録などを行うことができます。



4.スーパーコンピュータへの接続

以下のページを参考にスーパーコンピュータへの接続を行ってください。

システムへの接続方法

講習会アカウントの有効化

講習会アカウントの有効化について

講習会アカウントは、https://web.kudpc.kyoto-u.ac.jp/portal/guest/short_course_activation から、ご自身で有効化いただく必要があります。
講習会アカウントの発行・有効化には、プログラム講習会(Zoom Webiner)に参加した際にチャットでお知らせする 「照合コード」が必要 となります。
※ Webブラウザは **Google Chrome**を推奨しております。

講習会アカウントの有効化、システムへのログイン方法(動画マニュアル)