

豊橋技科大HPCクラスターの使い方

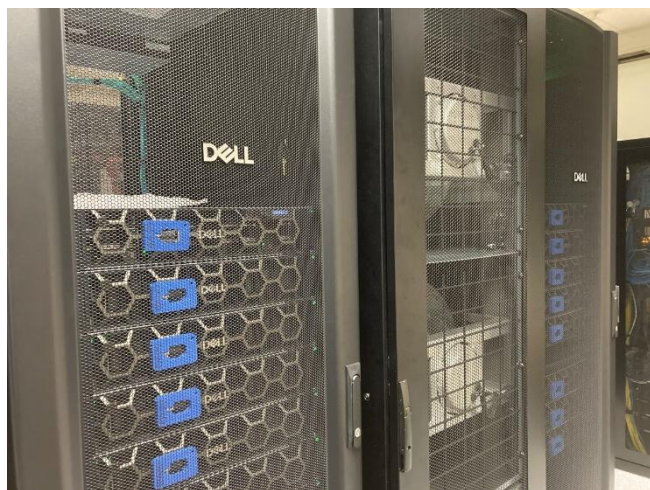
豊橋技術科学大学

情報メディア基盤センター



はじめに

- 豊橋技科大には共用のHPCクラスタが設置されています
HPC = High Performance Computing (高性能計算)



- 豊橋技科大の教員，学生は誰でも利用可能です
- 全国の国立高専，長岡技科大など連携機関からも利用可
<https://hpcportal.imc.tut.ac.jp/>

HPCクラスタの用途

- ❑ Linux環境での科学技術計算, シミュレーション, 深層学習などの実行
- ❑ 科学技術計算やシミュレーションのプログラム開発, プログラミングの学習にも利用可能です
- ❑ プログラミング (C/C++, Fortran)
コンパイラ : GNU, Intel, AMD, NVIDIA
並列計算ライブラリ : Intel MPI, Open MPI
数値演算ライブラリ : Intel MKL, AMD AOCL
- ❑ プログラミング : Python, Java, Perl, Ruby
- ❑ データ保存
/home/[数字]/[ID] 上限**20GB**
/work/[ID] 研究登録ユーザのみ利用可

計算機利用申込書の提出

- ❑ 登録料1,000円で研究ユーザとなり，HPCクラスタの利用制限が解除されます

<https://imc.tut.ac.jp/research/form>

- ❑ 研究用アプリケーション（2025年度）
HPCクラスタで利用する場合は計算機利用申込書の提出をお願いします

	用途
ABAQUS	有限要素解析
ANSYS HFSS	電磁界解析
COMSOL Multiphysics	有限要素解析
Gaussian	量子化学計算
Materials Studio	第一原理計算，量子化学計算

窓口サーバ

- HPCクラスタを使用するには、まず窓口サーバに接続してください
- プログラムのコンパイル、教育用アプリケーションの実行などが可能です
- 窓口サーバのアプリケーションは[こちらを参照](#)
- 窓口サーバの仕様は以下の通りです

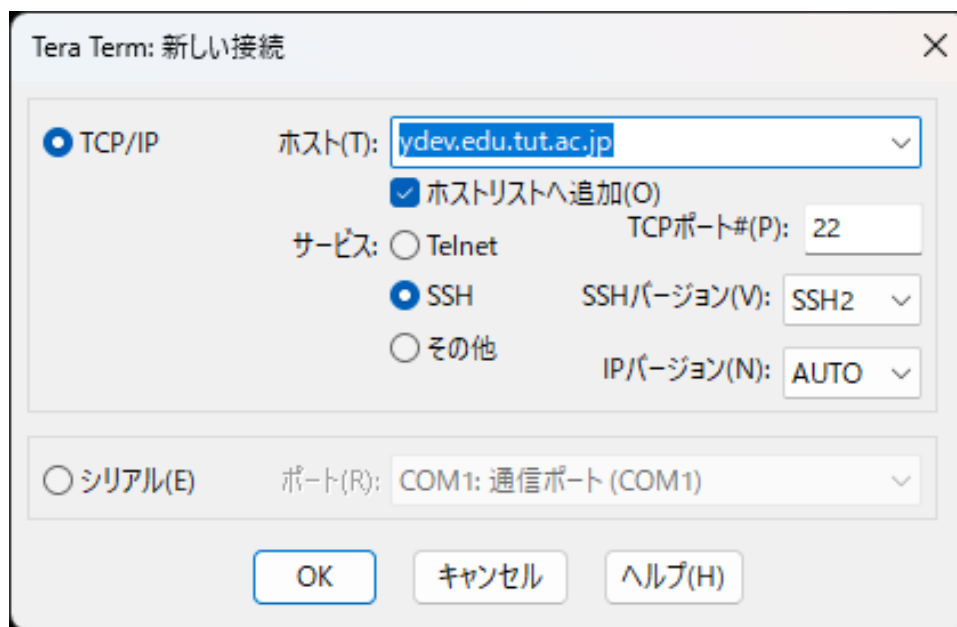
項目	仕様
ノード数	10
OS	RedHat Enterprise Linux 9.5
CPU	Intel Xeon Gold 6430 2.10 GHz
1ノードあたり CPUコア数	8
メモリ	24 GB

窓口サーバへの接続方法

- ❑ 学内ネットワークからの接続
パスワード認証または公開鍵認証によるSSH接続
- ❑ 学外ネットワークからの接続
公開鍵認証によるSSH接続
- ❑ 情報メディア基盤センターを利用するユーザ名とパスワードがあれば、誰でもログインし利用できます
- ❑ 本資料では Windows PC からTeraTermとWinSCPを用いて接続する方法を説明します

TeraTermによるログイン

1. TeraTermを起動 または ファイル → 新しい接続
2. ホストに **ydev.edu.tut.ac.jp** を指定
(豊橋技科大以外の利用者は **lark.imc.tut.ac.jp**)
サービスはSSH2, TCPポート#は22でOKをクリック



3. 初回のログインでは警告が表示されるが“続行”

TeraTermによるログイン

4. ユーザ名, パスフレーズを入力, 認証方式を指定
→ “OK”をクリックしてログイン

SSH認証

ログイン中: ydev.edu.tut.ac.jp

認証が必要です。

ユーザ名(N):

パスフレーズ(P):

☒ パスワードをメモリ上に記憶する(M)

☐ エージェント転送する(O)

認証方式

☒ ブレインパスワードを使う(L)

☐ RSA/DSA/ECDSA/ED25519鍵を使う

秘密鍵(K):

☐ rhosts(SSH1)を使う

ローカルのユーザ名(U):

ホスト鍵(F):

☐ キーボードインタラクティブ認証を使う(I)

☐ Pageantを使う

OK 接続断(D)

□ ユーザ名

情報メディア基盤センターから
発行されたアカウントの
ユーザ名を入力

□ パスフレーズ

上記アカウントのパスワード
または秘密鍵のパスフレーズ

□ 認証方式

“ブレインパスワードを使う”
または
“RSA/DSA/ECDSA/ED25519
鍵を使う”から秘密鍵の
ファイルを指定

TeraTermによる鍵生成と公開鍵の登録

1. TeraTermを起動し"新しい接続"ウィンドウ右上の×印をクリックして閉じる
2. 設定 → SSH鍵生成
3. ED25519を選択 → 生成をクリック
4. パスフレーズを設定して公開鍵, 秘密鍵を保存



The screenshot shows the 'TTSSH: 鍵生成' (TTSSH: Key Generation) dialog box. It features a green title bar with a close button (X) in the top right corner. The main area is divided into several sections: '鍵の種類' (Key Type) with radio buttons for RSA1, RSA, DSA, ECDSA-256, ECDSA-384, ECDSA-521, and ED25519 (which is selected); 'ビット数(B):' (Bit Count) set to 256; a '生成(G)' (Generate) button; a '閉じる(C)' (Close) button; and a 'ヘルプ(H)' (Help) button. Below these are input fields for '鍵のパスフレーズ:' (Key Password), 'パスフレーズの確認:' (Confirm Password), and 'コメント(O):' (Comment). At the bottom, there is a checked checkbox for 'bcrypt KDF形式(K)' (bcrypt KDF Format) and a 'ラウンド数(N):' (Number of Rounds) field set to 16. Finally, there are two buttons at the bottom: '公開鍵の保存(I)' (Save Public Key) and '秘密鍵の保存(P)' (Save Private Key).

TeraTermによる鍵生成

5. 以下のサイトの「プロフィールメンテナンス」→
「SSH公開鍵 - SSH Public Key」に公開鍵の内容を
(コピーアンドペーストで) 登録

<https://imc.tut.ac.jp/config/>

豊橋技科大以外の利用者は

<https://hpcportal.imc.tut.ac.jp/>
より“プロフィール変更”

秘密鍵は他人に知られない
よう十分注意してください



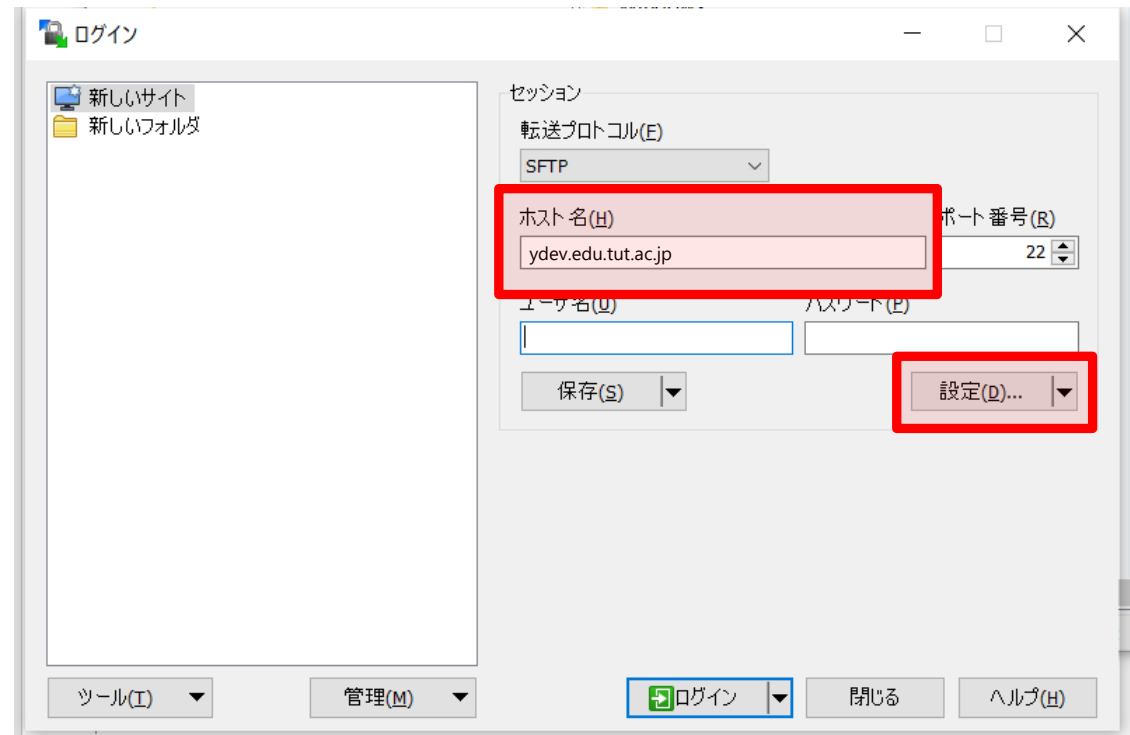
The screenshot shows a web form for SSH Public Key registration. The title is "SSH公開鍵 - SSH Public Key". The form contains several input fields for user identification: VLAN ID, ORCID ID, Kaken Researcher Number, ResearchMap Login ID, Scopus Author ID, and Web Of Science Researcher Identifier. There is also a field for the password recovery email address. At the bottom, there are "保存" (Save) and "リセット" (Reset) buttons. A red box highlights the top section of the form, which includes the title and the SSH Public Key input field.

SSH公開鍵 - SSH Public Key	
VLAN ID - VLAN ID	
ORCID識別子 - ORCID ID	
科研費研究者番号 - Kaken Researcher Number	
ResearchMapログイン名 - Research Map Login Id	
Scopus著者ID - Scopus Author Identification Number	
WebOfScience著者識別子 - Web Of Science Researcher Identifier	
(※英文字1文字-数字4桁-数字4桁の形式。 例) a-1234-5678)	
パスワード復旧用メールアドレス - Mail Address for Password Recovery	

保存 リセット

WinSCPによるファイル転送

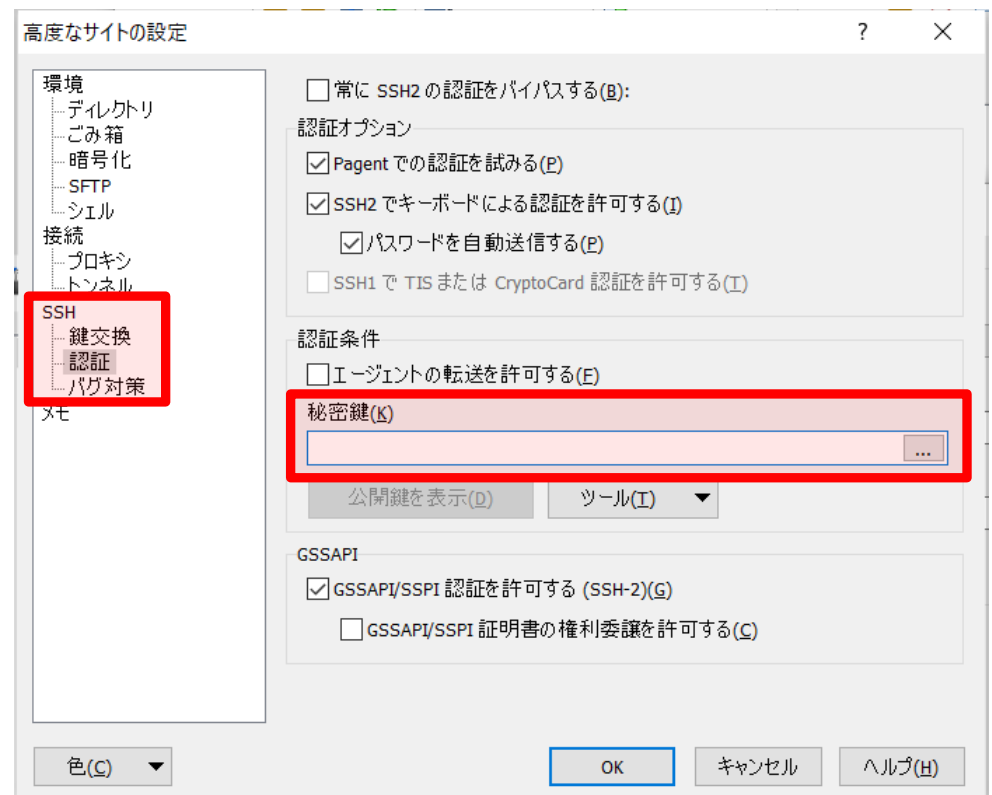
1. WinSCPを起動
2. 新しいサイト → ホスト名に **ydev.edu.tut.ac.jp**
(**lark.imc.tut.ac.jp**) を入力
3. ユーザ名, パスワードを入力してログイン
4. 公開鍵認証の場合は“設定”をクリック



WinSCPによるファイル転送

5. SSH → 認証 から秘密鍵を指定
6. OK → ログイン でWindows PCとファイルのやりとりができます

ログインの前に
"保存"をクリックして
ログイン情報を登録
しておくと便利です



窓口サーバへの接続について

- TeraTermとWinSCPは情報メディア基盤センターの教育用端末にインストールされています
- 各種OSにインストールされている「ターミナル」やVS Codeのターミナルなどからも接続可能です
接続のコマンド：`ssh [ID]@ydev.edu.tut.ac.jp`
- GUIアプリケーションを利用する際には
リモートデスクトップ接続も有効です
- 接続方法についてはこちらもご参照ください
<https://hpcportal.imc.tut.ac.jp/wiki/SSHClient/>

Linuxのコマンド

❑ TeraTermなどでログインしてからは
Linuxのコマンドで操作します

コマンド	説明
ls [ファイル/ディレクトリ]	ファイル/ディレクトリ情報の表示
cd [ディレクトリ]	ディレクトリの移動
cd ..	一つ上のディレクトリに移動
cd ~	ホームディレクトリに移動
mkdir [ディレクトリ]	ディレクトリの作成
rmdir [ディレクトリ]	ディレクトリの削除 ※ 中身が空になっている必要あり
mv [変更前ディレクトリ] [変更後ディレクトリ]	ディレクトリ名の変更
vi [ファイル名]	ファイルの編集 ※ Escape → :q! → Enter で終了
emacs [ファイル名]	ファイルの編集 ※ Ctrl+x → Ctrl+c で終了

Linuxのコマンド

コマンド	説明
ls [ファイル or ディレクトリ]	ファイルまたはディレクトリ情報の表示
less [ファイル]	ファイルの表示 ※ space, b, カーソルキーで操作, q で終了
cp [ファイル] [ディレクトリ]	ファイルを指定したディレクトリにコピー
cp [ファイル1] [ファイル2]	ファイル1と同内容のファイル2を生成
mv [ファイル] [ディレクトリ]	ファイルを指定したディレクトリに移動
mv [変更前ファイル] [変更後ファイル]	ファイル名の変更
rm [ファイル]	ファイルの削除
cat *.f90 > temp.f90	ファイルの出力 この場合, 拡張子がf90のすべてのファイルをtemp.f90に出力する
grep "temp" *.f90	ファイル中の文字列の検索 この場合, 拡張子がf90のすべてのファイルを対象として"temp"を含む行を表示
logout	ログアウト ※ Ctrl + d でも可

Emacsの操作

コマンド	説明
Ctrl+x → Ctrl+s	上書き保存
Ctrl+x → Ctrl+w	別名で保存
Ctrl+x → Ctrl+c	Emacsの終了
Ctrl+x → u	元に戻す
Ctrl+r	前方検索
Ctrl+s	後方検索
Meta (Escape) → <	冒頭へ移動
Meta (Escape) → >	末尾へ移動
Meta (Escape) → gg → Enter	指定した行番号へ移動
Ctrl+space	領域選択
Ctrl+w	指定した領域を切り取り
Meta (Escape) → w	指定した領域をコピー
Ctrl+y	貼り付け

module コマンド

- ❑ コンパイラや研究用アプリケーションを使用するために **module コマンド** が用意されています
- ❑ 窓口サーバ, 計算サーバで利用できます
- ❑ 利用可能なモジュールを表示
`$ module avail`
- ❑ ロードしたモジュールを表示
`$ module list`
- ❑ module コマンドによる設定を解除
`$ module unload`
- ❑ 計算サーバでは以下のコマンドで module コマンドが有効になります
`$ source /etc/profile`

module コマンド

- ❑ Intelコンパイラを利用する場合

```
$ module load intel/2025
```

- ❑ Intel MPIを利用する場合

```
$ module load intelmpi.intel/2025
```

- ❑ Open MPIを利用する場合

```
$ module load openmpi5.04
```

- ❑ MPI環境は競合するためどちらかのみロードすること

- ❑ 計算サーバでもモジュールを利用できます

```
source /etc/profile としてから module コマンドを  
実行してください
```

C言語プログラムのコンパイル

□ GNUコンパイラ

□ 逐次実行

```
gcc -o temp.x temp.c
```

□ OpenMP並列

```
gcc -fopenmp -o temp.x temp.c
```

□ MPI並列 (Open MPI)

```
env OMPI_CC=gcc mpicc -o temp.x temp.c
```

□ Intelコンパイラ

□ 逐次実行

```
icx -o temp.x temp.c
```

□ OpenMP並列

```
icx -qopenmp -o temp.x temp.c
```

□ MPI並列 (Intel MPI)

```
mpicc -cc=icx -o temp.x temp.c
```

C++プログラムのコンパイル

□ GNUコンパイラ

□ 逐次実行

```
g++ -o temp.x temp.cpp
```

□ OpenMP並列

```
g++ -fopenmp -o temp.x temp.cpp
```

□ MPI並列 (Open MPI)

```
env OMPI_CXX=g++ mpicxx -o temp.x temp.cpp
```

□ Intelコンパイラ

□ 逐次実行

```
icpx -o temp.x temp.cpp
```

□ OpenMP並列

```
icpx -qopenmp -o temp.x temp.cpp
```

□ MPI並列 (Intel MPI)

```
mpicxx -cxx=icpx -o temp.x temp.cpp
```

Fortranプログラムのコンパイル

□ GNUコンパイラ

□ 逐次実行

```
gfortran -o temp.x temp.f90
```

□ OpenMP並列

```
gfortran -fopenmp -o temp.x temp.f90
```

□ MPI並列 (Open MPI)

```
env OMPI_FC=gfortran mpif90 -o temp.x temp.f90
```

□ Intelコンパイラ

□ 逐次実行

```
ifx -o temp.x temp.f90
```

□ OpenMP並列

```
ifx -qopenmp -o temp.x temp.f90
```

□ MPI並列 (Intel MPI)

```
mpiifort -fc=ifx -o temp.x temp.f90
```

コンパイラオプション

❑ `gcc --help | less` などと入力して確認できます

❑ 代表的なオプションは以下の通り

オプション	説明
<code>-o</code>	実行ファイル名を指定
<code>-O0</code> , <code>-O1</code> , <code>-O2</code> , <code>-O3</code>	コンパイラによる最適化によりプログラムを高速化 <code>-O0</code> は最適化しない, <code>-O3</code> が最速
<code>-fopenmp</code> (GNU compiler) <code>-qopenmp</code> (Intel compiler)	OpenMPによる並列化
<code>-fbacktrace -g</code> (GNU) <code>-traceback -g</code> (Intel)	デバッグ用オプション 実行時にデバッグ情報を生成する
<code>-fcheck=all</code> (GNU) <code>-check all</code> (Intel)	デバッグ用オプション 実行時にチェックを行いエラーや警告を出力する
<code>-qmkl</code> (Intel)	Intel Math Kernel Library (MKL) を使用する BLAS, LAPACKなどのライブラリを利用可能
<code>-static</code>	ライブラリを静的リンクする. ライブラリがインストールされていない環境でもプログラムを実行できるようになる

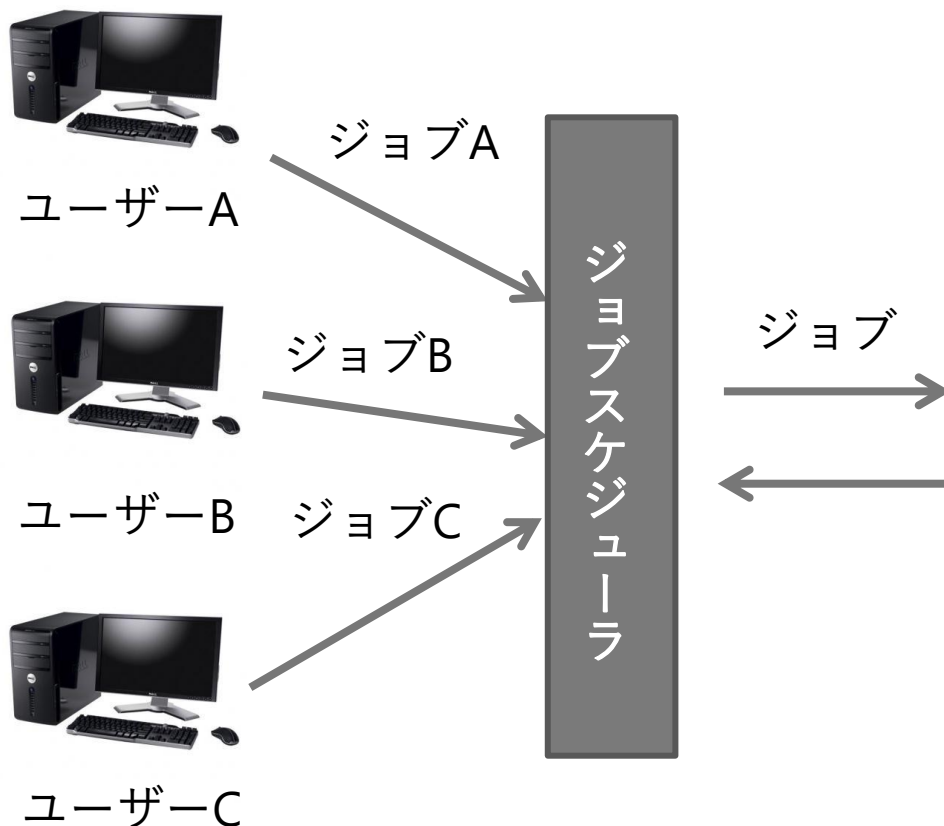
計算サーバ

- ❑ 複数CPUコアを長時間占有する処理は計算サーバで行ってください
- ❑ このような処理を窓口サーバで行うと他の利用者の迷惑となります
- ❑ 計算サーバは全部で15ノードあります。共用の計算資源であることに留意して教育・研究にご活用ください
- ❑ 計算サーバ（1ノード）の仕様は以下の通りです

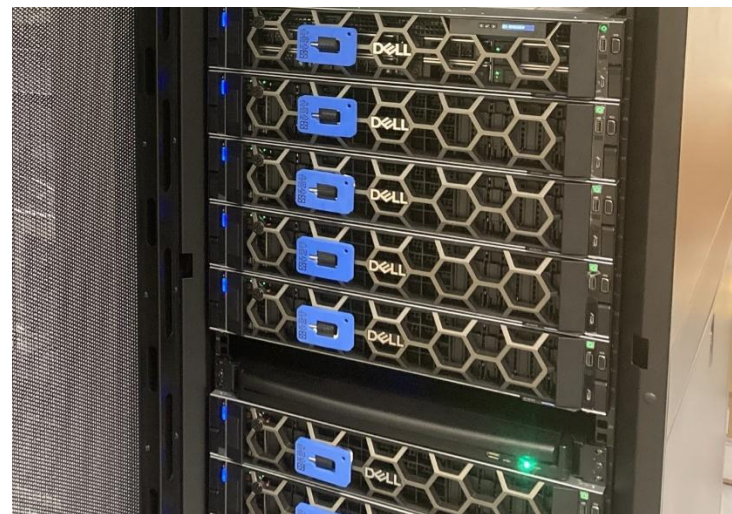
項目	仕様
機種名	DELL PowerEdge R7625
OS	RedHat Enterprise Linux 9.5
CPU	AMD EPYC 9254 2.90 GHz 24コア x 2
メモリ	256 GB
GPU	AMD MI210 PCIe (64 GB) x 2

ジョブスケジューラ

- 複数のユーザが共同で計算資源を利用
→ ジョブスケジューラ PBS Professional による管理



計算サーバ
全15ノード
ysnd00~ysnd14



ジョブスケジューラの利用方法

- ❑ 事前にジョブを実行するためのスクリプトファイル (bashなど) を用意
- ❑ ジョブ投入
\$ **qsub** [実行スクリプトファイル名]
- ❑ ジョブの状態, ジョブID表示
\$ qstat -a
- ❑ ジョブの詳細を表示
\$ qstat -f | less
- ❑ ジョブの削除
\$ qdel [ジョブID]
- ❑ ジョブが実行されると標準出力と標準エラー出力が別々のファイルに出力されます

ジョブスケジューラの利用方法

□ qsub コマンドのオプション

オプション	使用例	意味
-q	-q Eduq	ジョブを投入するキューを指定
-v	-v SINGULARITY_ IMAGE=***.sif	指定したSingularityイメージ上でジョブを実行
-l	-l mem=1g	使用するCPUコア数, メモリ上限などを設定
-o	-o filename	標準出力の内容を指定されたファイルに出力
-e	-e filename	標準エラー出力の内容を指定されたファイルに出力
-j	-j oe	標準エラー出力を標準出力にマージ
-m	-m e	ジョブ終了時(e)など, 指定したタイミングでメール送信
-M	-M ***@tut.jp	メールの送信先

□ これらは実行スクリプトファイル内の指示文でも指定できます

利用可能なキュー

□ 以下のキューは全員が利用できます

キュー	割当 ノード数	CPUコア数 最大/標準	メモリ 最大/標準	GPU数 最大/標準	経過時間 上限	備考
Eduq	15	8/1	64GiB/4GiB	2/0	8時間	AMD GPUを 利用可能
iEduq	2	8/1	64GiB/4GiB	1/0	8時間	NVIDIA GPUを 利用可能

□ これらのキューでは[インタラクティブジョブ](#)により対話処理を実行できます

□ インタラクティブジョブを利用してGPUを使用するプログラムのコンパイルが可能です。詳しくは[こちら](#)を参照してください

□ これらのキューは同時に実行できるジョブは1つのみです

利用可能なキュー

□ 計算機利用申込書の提出により以下のキューも利用できます

キュー	割当 ノード数	CPUコア数 最大/標準	メモリ 最大/標準	GPU数 最大/標準	経過時間 上限	備考
sRchq	13	48/1	256GiB/8GiB	2/0	336時間	優先して実行
mRchq	13	384/1	2048GiB/6GiB	2/0	336時間	
iRchq	2	96/1	512GiB/8GiB	0/0	336時間	

- これらはインタラクティブジョブは不可（バッチジョブのみ）
- 学生が計算機利用申込書を提出する際には、事前に指導教員に相談をお願いします

コンテナイメージ

- ❑ HPCクラスタにおけるジョブはSingularityコンテナによる仮想環境で実行されます
- ❑ 特に指定のない場合、コンテナイメージは自動的に指定されます
- ❑ コンテナイメージを指定する場合のオプション
-v SINGULARITY_IMAGE=<イメージ名>.sif
- ❑ 以下のイメージが用意されています
/common/Singularity_sif/prg_env_amd_2025.01.sif
/common/Singularity_sif/prg_env_intel_2025.01.sif
- ❑ ユーザが用意したイメージファイルを指定することもできます
- ❑ イメージファイルの取得については[こちら](#)

実行スクリプトファイルの例：逐次プログラム

```
#!/bin/bash
```

```
#PBS -q Eduq 教育用途で使用可能なキュー
```

```
#PBS -l select=1:ncpus=1:vnode=ysnd01
```

```
(空行)
```

ysnd01の1 CPUコアを使用

```
ulimit -c 0
```

```
source /etc/profile      moduleコマンドを利用可能にしてから
```

```
module load intel/2025  Intelコンパイラ, ライブラリを有効化
```

```
cd $PBS_O_WORKDIR      実行スクリプトのディレクトリに移動
```

```
./test.x
```

- 上記をtest.bashとして保存し, \$ qsub test.bash でtest.xを実行できます

実行スクリプトファイルの例：OpenMP並列

```
#!/bin/bash
```

```
#PBS -q Eduq
```

ysnd01の 4 CPUコアを使用

```
#PBS -l select=1:ncpus=4:vnode=ysnd01
```

(空行)

```
ulimit -c 0
```

```
source /etc/profile
```

```
module load intel/2025
```

```
export OMP_NUM_THREADS=4
```

OpenMPによる 4 スレッド並列

```
cd $PBS_O_WORKDIR
```

```
./test.x
```

実行スクリプトファイルの例：MPI並列

```
#!/bin/bash
#PBS -q Eduq
#PBS -l select=1:ncpus=4:mpiprocs=4
(空行)
ulimit -c 0
source /etc/profile
module load intel/2025
module load intelmpi/2025
cd $PBS_O_WORKDIR
export OMP_NUM_THREADS=1
mpirun -np 4 ./test.x
```

1ノード,
4 CPUコア, 4 MPIプロセス

MPIのライブラリをロード

4 MPIプロセスでプログラムを実行

※ 複数ノードでMPI並列を行う場合は[こちら](#)を参照

実行スクリプトファイルの例：MPI/OpenMPハイブリッド並列

```
#!/bin/bash
#PBS -q Eduq
#PBS -l select=1:ncpus=4:mpiprocs=2
(空行)
ulimit -c 0
source /etc/profile
module load intel/2025
module load intelmpi/2025
export OMP_NUM_THREADS=2
cd $PBS_O_WORKDIR
mpirun -np 2 ./test.x
```

1ノード,
1ノードあたり4 CPUコア,
1ノードあたり2 MPIプロセス

OpenMPにより1プロセスあたり
2 スレッド並列

2 MPIプロセスでプログラムを実行

実行スクリプトファイルの例：計算資源の設定

```
#!/bin/bash
```

```
#PBS -q sRchq
```

```
#PBS -l walltime=24:00:00 ← 経過時間の上限
```

```
#PBS -l select=1:vnode=ysnd06:ncpus=4:mem=48GB
```

ysnd06の4 CPUコア，メモリ48 GBを占有

- ❑ Eduq では 計算サーバ（vnode）として ysnd00～ysnd14 を指定できます。ysnd00, ysnd01はEduq専用ですので，積極的に利用してください
- ❑ sRchq, mRchq では ysnd02～ysnd14 を指定できます
- ❑ あらかじめ計算時間，必要なメモリ量を見積って効率よく利用しましょう

インタラクティブジョブによるGPUプログラミング

窓口サーバにログインして

```
$ qsub -I -q Eduq -l select=1:ncpus=1:ngpus=1 -v  
SINGULARITY_IMAGE=/common/Singularity_sif/prg_env_amd_2025  
.01.sif
```

と入力，実行するとインタラクティブジョブによって計算サーバで対話処理ができます．

計算サーバ(ysnd00～ysnd14)にはAMD MI210 GPUが搭載されています．CPUやGPUの情報は以下のコマンドで確認できます．

```
$ rocminfo
```

AMD GPUで動作するコードを以下のコマンドでコンパイルできます．

```
$ hipcc -I/opt/rocm/include -D__HIP_PLATFORM_AMD__ [ソース  
ファイル名] -o [出力するプログラム名]
```

インタラクティブジョブによるGPUプログラミング

NVIDIAのGPUを利用する場合は

```
$ qsub -I -q iEduq -l select=1:ncpus=1:ngpus=1 -v  
SINGULARITY_IMAGE=/common/Singularity_sif/prg_env_intel_20  
25.01.sif
```

と入力，実行します．

IoT・AI基盤システム(yind00, yind01)にはA100 GPUが搭載されています．GPUの情報は以下のコマンドで確認できます．

```
$ nvidia-smi
```

NVIDIA GPUで動作するC++のコードを以下のコマンドでコンパイルできます．

```
$ nvcc -ccbin g++ -I/usr/include [ソースファイル名] -o [出力  
するプログラム名]
```

注意事項

- ❑ プログラムがエラーで終了した際に生成される **core ファイル** に注意してください
- ❑ core ファイルによりユーザに割り当てられた容量をオーバーし、作業が継続できなくなる場合があります
- ❑ すでに生成された core ファイルは削除してください
- ❑ "**ulimit -c 0**" のコマンドにより core ファイルを生成しないようにできます
- ❑ ファイルを削除しても作業が継続できなくなった場合は情報メディア基盤センタ (supports@imc.tut.ac.jp) までお知らせください
- ❑ リモートデスクトップ接続 で接続、利用した後はログアウトをお願いします

謝辞について

□ HPCクラスタを使用して得た成果を論文，学会等で発表する際には謝辞（Acknowledgments）に記載をお願い致します

□ 例

本研究で行った計算は豊橋技術科学大学情報メディア基盤センターのHPCクラスタを利用して行いました。

□ Example

The authors would like to thank Information and Media Center (IMC) at Toyohashi University of Technology for providing computer resources (HPC cluster).