

FOCUS スパコン

利用の手引き

計算科学振興財団

目次

1.	FOCUS スパコンシステムの概要	5
1.1.	システム構成	5
1.1.1.	システム概要図	5
1.1.2.	システム構成要素の特徴	6
1.2.	ハードウェア構成	8
1.2.1.	サーバ	8
1.3.	ソフトウェア構成	11
1.3.1.	オペレーティングシステム (OS)	11
1.3.2.	ソフトウェア	12
1.3.3.	アプリケーション	13
1.4.	製品マニュアル	14
1.5.	ディレクトリ構成	15
1.5.1.	共用フロントエンドサーバ	15
1.5.2.	演算ノード	15
2.	システムの使用方法	16
2.1.	システムへのログイン	16
2.1.1.	インターネットからの SSH 接続によるログイン	16
2.1.2.	インターネットからの SSL-VPN 接続による利用方法	35
2.1.3.	高度計算科学研究支援センター内でのログイン方法	38
2.2.	パスワードの変更	39
2.2.1.	パスワードの変更 (センター外)	39
2.2.2.	パスワードの変更 (センター内)	39
2.3.	ログインシェル	40
2.4.	追加ストレージ領域 (/home3, Luster File System) の利用方法	41
2.4.1.	Lustre File System 環境構成概要	42
2.4.2.	/home3 利用状況の確認	42
2.4.3.	Stripe Size/Stripe Count	43
2.5.	クラウドストレージの利用方法 (提供終了)	47
2.6.	改行コード	48
2.6.1.	改行コード	48
2.6.2.	エディタ	48
2.6.3.	改行コードの変換	48
2.7.	module コマンド	49
3.	コンパイラ、MPI の使用方法	51
3.1.	Intel コンパイラ	51
3.1.1.	コンパイラ環境の設定 (Intel コンパイラ)	51
3.1.2.	コンパイルコマンド (Intel コンパイラ)	51
3.1.3.	MPI 環境の設定 (Intel コンパイラ)	52
3.1.4.	コンパイラ、MPI 環境の切替え	53
3.1.5.	コンパイル・オプション (Intel コンパイラ)	54
3.1.6.	コンパイル方法 (Intel コンパイラ)	56
3.1.7.	コンパイル時の注意点 (Intel コンパイラ)	57
3.2.	GNU コンパイラ	58

3.2.1.	コンパイラ環境変数の設定 (GNU コンパイラ)	58
3.2.2.	コンパイルコマンド (GNU コンパイラ)	58
3.2.3.	MPI 環境変数の設定 (GNU コンパイラ)	59
3.2.4.	コンパイラ、MPI 環境の切替え	60
3.2.5.	コンパイル・オプション (GNU コンパイラ)	61
3.2.6.	コンパイル方法 (GNU コンパイラ)	62
3.2.7.	コンパイル時の注意点 (GNU コンパイラ)	63
4.	プログラムの実行方法	64
4.1.	キュー	64
4.1.1.	キューの一覧	64
4.1.2.	キュー情報の確認方法	66
4.1.3.	利用可能なノード数の確認方法	68
4.2.	ジョブの実行方法 (SLURM コマンド編)	69
4.2.1.	ジョブ投入コマンド (sbatch)	69
4.2.2.	ジョブ情報表示コマンド (squeue)	70
4.2.3.	ジョブのキャンセルコマンド (scancel)	71
4.2.4.	ジョブステータス情報表示コマンド (sstat)	71
4.2.5.	実行ジョブ情報表示コマンド (sacct)	73
4.3.	ジョブの実行方法 (fj コマンド編)	74
4.3.1.	ジョブ投入コマンド (fjsub)	74
4.3.2.	ジョブ情報表示コマンド (fjstat)	75
4.3.3.	ジョブのキャンセルコマンド (fjdel)	76
4.4.	ジョブ投入スクリプトの作成	77
4.4.1.	処理方法の指定	77
4.4.2.	sbatch オプション	77
4.4.3.	環境変数	78
4.4.4.	逐次ジョブを実行する場合	78
4.4.5.	スレッド並列ジョブを実行する場合	79
4.4.6.	MPI プログラム (OpenMPI) を実行する場合	80
4.4.7.	MPI プログラム (Intel MPI) を実行する場合	81
4.4.8.	MPI プログラム (mpich2) を実行する場合	81
4.5.	Xeon Phi コプロセッサの使用	82
4.6.	課金確認コマンド	83
4.6.1.	プロジェクト単位従量課金確認コマンド thismonth	83
4.6.2.	利用者単位従量課金確認コマンド uacct	85
4.6.3.	利用者単位アプリケーション課金確認コマンド uacct_apl	87
4.6.4.	課金確認コマンドの情報反映タイミング	89
4.7.	実行ジョブ一覧の確認方法	90
5.	インターネット高速転送システムの使用方法	92
5.1.	Skeed Silver Bullet の利用申請	92
5.2.	専用クライアント (SkeedSilverBullet GUI) の使用方法	93
5.2.1.	クライアントのインストール	93
5.2.2.	クライアントの環境設定	95
5.2.3.	クライアントの起動	97
5.2.4.	ファイルのアップロード	98
5.2.5.	ファイルのダウンロード	100

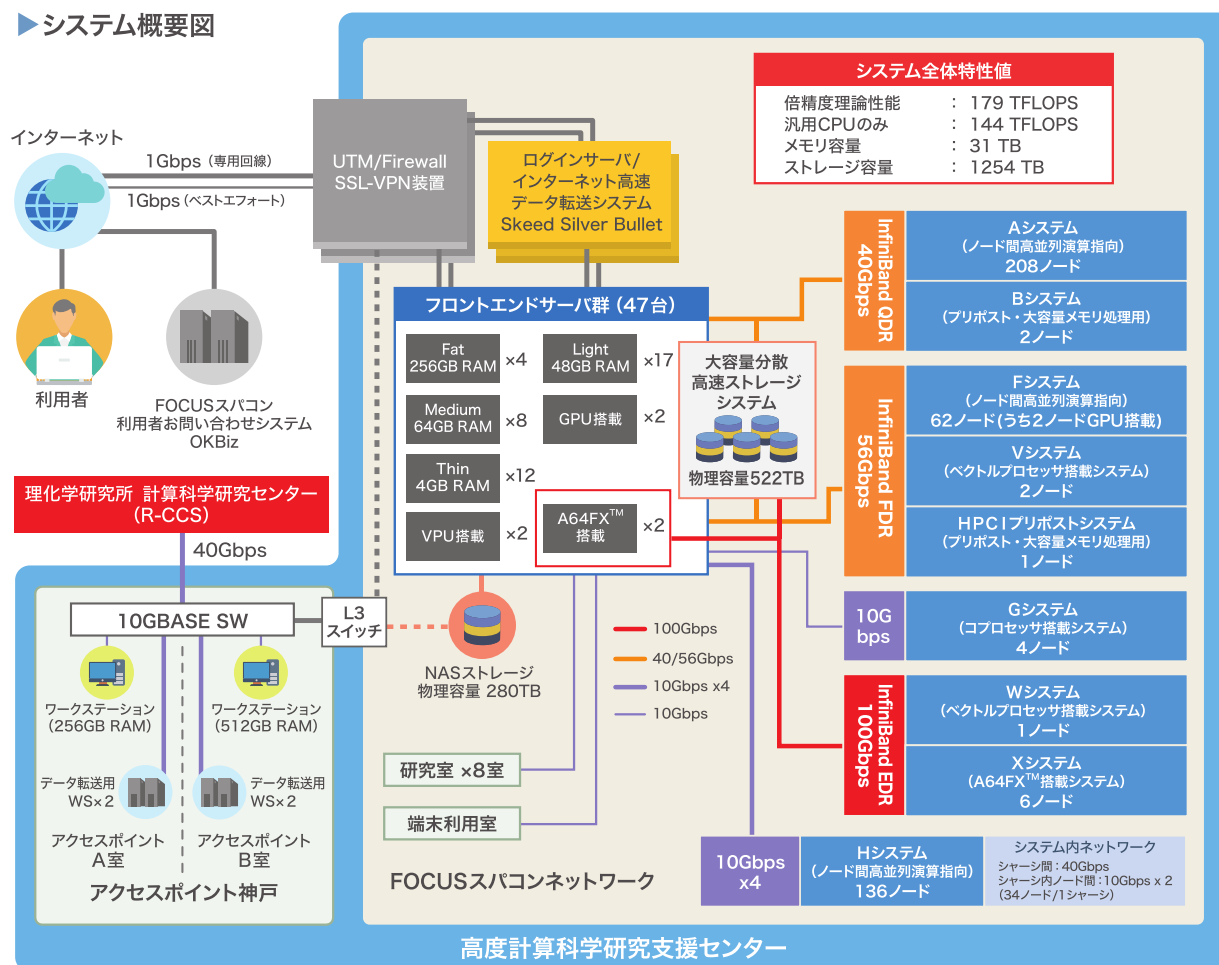
5.3. Web ブラウザベースの使用方法	102
5.3.1. システムへのログイン	102
5.3.2. ファイルのアップロード	104
5.3.3. ファイルのダウンロード	105
5.3.4. ファイルの削除	106
5.3.5. Web ブラウザーからの専用クライアント起動	108
付録 A. FOCUS スパコンシステム各種サーバ・ストレージ概要	110
付録 B. コマンド比較表 (SLURM と LSF)	111
付録 C. プログラムとジョブ投入スクリプトのサンプル	113

1. FOCUS スパコンシステムの概要

1.1. システム構成

1.1.1. システム概要図

▶ システム概要図



1.1.2. システム構成要素の特徴

FOCUS スパコンシステムを構成する要素について、特徴は以下の通りです。

- **ファイアウォール**

インターネットからの不正アクセスを防ぎます。各種ポートを塞いだり一方通行にしたり等のアクセス制御を行います。

- **ウェブサーバシステム**

FOCUS スパコンシステムの利用に関する情報を提供します。運用情報や予約状況等を表示します。

- **ログインサーバシステム**

インターネットからの SSH プロトコルを使用したログイン接続、ファイル転送の中継を行います。この計算機に一旦ログインしてから再度、フロントエンドサーバシステムに SSH でログイン、ファイル転送を行います。

- **フロントエンドサーバシステム**

FOCUS スパコンシステムを利用するための拠点となります。プロジェクト毎の専用のファイルシステムがマウントされ、利用者がログインし、プログラムの開発、ジョブ管理システムの利用、小規模な解析・デバッグ、小規模なプリポスト処理等を行います。

GPU NVIDIA® Quadro® P1000 を利用できるサーバもあります。

SX-Aurora TSUBASA を利用できるサーバもあります。

共用フロントエンドサーバの利用については

「2.1.1.3. 共用フロントエンドサーバへの接続」に記載の

【共用フロントエンド利用についての注意点】をご確認ください。

- **A システム** ノード間高並列演算指向

ノード内 12 コアを利用した共有メモリ並列から、Infiniband-QDR (40Gbit/s) で接続されたノード間分散メモリ並列、それらを組み合わせたハイブリッド並列にご利用いただけます。

- **B システム** プリポスト・大容量メモリ処理用

大容量共有メモリ (512GB) を備え、入力データを作成したり、計算した結果をディスク上から大規模共有メモリに読込んで高速にポスト処理を行ったりするためのシステムです。ノード内に 16 コアを備えております。

- **D システム** ノード内並列／ノード間高並列演算兼用

ノード内 20 コアを利用した共有メモリ並列から、Infiniband-FDR (56Gbit/s) で接続されたノード間分散メモリ並列、それらを組み合わせたハイブリッド並列にご利用いただけます。

- **F システム** ノード内並列／ノード間高並列演算兼用

ノード内 40 コアを利用した共有メモリ並列から、Infiniband-FDR (56Gbit/s) で接続されたノード間分散メモリ並列、それらを組み合わせたハイブリッド並列にご利用いただけます。

GPU NVIDIA® Tesla® P100 を利用できるノードもあります。

- **G システム** コプロセッサ搭載システム

インテル製 「Xeon Phi 5110P」 (1 基あたり 60 コア) をノード 1 台あたり 1 基搭載しております。ノード内 12 コアを利用した共有メモリ並列から、10 Gigabit Ethernet (10Gbps) で接続されたノード間分散メモリ並列、それらを組み合わせたハイブリッド並列にご利用いただけます。

- **H システム** ノード間高並列演算兼用

ノード内 8 コアを利用した共有メモリ並列から、10 Gigabit Ethernet (10Gbps) で接続されたノード間分散メモリ並列、それらを組み合わせたハイブリッド並列にご利用いただけます。

・**V システム** ベクトルエンジン搭載システム

NEC 製ベクトルエンジン「SX-Aurora TSUBASA Type 10B」をノード 1 台あたり 1 基搭載しております。また、ノード内 20 コアを利用した共有メモリ並列から、Infiniband-FDR (56Gbps) で接続されたノード間分散メモリ並列、それらを組み合わせたハイブリッド並列にご利用いただけます。

・**W システム** ベクトルエンジン搭載システム

NEC 製ベクトルエンジン「SX-Aurora TSUBASA Type 10B」を 8 基搭載しております。また、ノード内 40 コアを利用した共有メモリ並列にご利用いただけます。

・**X システム** A64FX 搭載システム

『富岳』と同じ FUJITSU 製 CPU「A64FX」を搭載しており、ノード内 48 コアを利用した共有メモリ並列から、Infiniband-EDR (100Gbps) で接続されたノード間分散メモリ並列、それらを組み合わせたハイブリッド並列にご利用いただけます。

・**HPCI プリポストシステム** プリポスト・大容量メモリ処理用

大容量共有メモリ (1.5TB) を備え、入力データを作成したり、計算した結果をディスク上から大規模共有メモリに読込んで高速にポスト処理を行ったりするためのシステムです。GPU NVIDIA Quadro P4000 も 2 基搭載しております。

・**NAS ストレージシステム** ホーム領域 (/home1)

物理容量は 360TB です。

/home1 としてマウントされております。

利用者のホーム領域として課題毎に 200GB の利用が可能です。通信帯域は全体で 500MB/s です。通信帯域は FOCUS スパコン全利用者にて共有しているため、I/O 性能は自身も含めた全利用者の I/O 状況に依存します。

・**大容量分散高速ストレージシステム** ホーム兼ワーク領域 (/home3)

物理容量は 1640TB です。

/home3 としてマウントされています。

利用者のホーム兼ワーク領域となり、利用には追加契約が必要です。

月単位・10GB 単位で容量を追加/削減することが可能です。契約容量を削減し、実利用量が契約容量を上回っている場合は利用分課金されます (10GB 単位) のでご注意ください。通信帯域は全体で 11GB/s、プロセスあたり最大 1GB/s です。分散ファイルシステムであり、前述 NAS ストレージシステムと比べて 22 倍の通信帯域となっておりますので、大規模 並列計算の複数プロセスによる I/O に適しています。

1.2. ハードウェア構成

1.2.1. サーバ

・A システム

ハードウェア	富士通 BX922 S2
CPU	Intel Xeon L5640 (2.26GHz) ×2CPU (計 12 コア) / ノード
メモリ	48 GB / ノード
インターフェース	Infiniband-QDR (40Gbps) ×1 / ノード
ターボブースト	ON

・B システム

ハードウェア	富士通製 RX600 S5
CPU	Intel Xeon E7520 (1.86GHz) ×4CPU (計 16 コア) / ノード
メモリ	512 GB / ノード
インターフェース	Infiniband-QDR (40Gbps) ×1 / ノード
ターボブースト	なし

・D システム

ハードウェア	Cray H2312WFFKR (HPC 専用 2U ラックマウント型 ブレードサーバ)
CPU	Intel Xeon E5-2670 v2 (2.5GHz) ×2CPU (計 20 コア) / ノード
メモリ	64 GB / ノード
インターフェース	Infiniband-FDR (56Gbps) ×1 / ノード
ターボブースト	ON

・F システム

ハードウェア	富士通製 CX2550M2
CPU	Intel Xeon E5-2698v4 (2.2GHz) ×2CPU (計 40 コア) / ノード
GPU	NVIDIA Tesla P100 16GB ×1 / ノード (2 ノードのみ搭載)
メモリ	128 GB / ノード
インターフェース	Infiniband-FDR (56Gbps) ×1 / ノード
ターボブースト	ON

・G システム

ハードウェア	NEC Express5800/HR120a-1
CPU	Intel Xeon E5-2640 (2.5GHz) ×2CPU (計 12 コア) / ノード
コプロセッサ	Intel Xeon Phi 5110P ×1 基 (計 60 コア) / ノード
メモリ	64 GB / ノード
インターフェース	10 Gigabit Ethernet (10Gpbs) ×1 / ノード
ターボブースト	ON

・H システム

ハードウェア	NEC DX2000 サーバモジュール
CPU	Intel Xeon D-1541 (2.10GHz) ×1CPU (計 8 コア) / ノード
メモリ	64 GB / ノード
インターフェース	10 Gigabit Ethernet (10Gpbs) ×1 / ノード

ターボブースト	ON
---------	----

・v システム

ハードウェア	NEC SX-Aurora TSUBASA A300-2
CPU	Intel Xeon Gold 6148 (2.40GHz) ×1CPU(計 20 コア)/ノード
ベクトルエンジン	SX-Aurora TSUBASA Type 10B (48GB) ×1/ノード
メモリ	96 GB/ノード
インターフェース	10 Gigabit Ethernet (10Gpbs) ×1/ノード Infiniband-FDR (56Gbps) ×1/ノード
ターボブースト	ON

・w システム

ハードウェア	NEC SX-Aurora TSUBASA A300-8
CPU	Intel Xeon Gold 6148 (2.40GHz) ×2CPU(計 40 コア)/ノード
ベクトルエンジン	SX-Aurora TSUBASA Type 10B (48GB) ×8/ノード
メモリ	192 GB/ノード
インターフェース	Infiniband-EDR (100Gbps) ×1/ノード
ターボブースト	ON

・HPCI プリポスト

ハードウェア	富士通製 RX2540 M4
CPU	Intel Xeon Silver 4112 (2.60GHz) ×2CPU(計 8 コア)/ノード
GPU	NVIDIA Quadro P4000 8GB ×2/ノード
メモリ	1.5 TB/ノード
インターフェース	10 Gigabit Ethernet (10Gpbs) ×4/ノード Infiniband-FDR (56Gbps) ×1/ノード
ターボブースト	ON

・共用フロントエンドサーバ

ハードウェア	Intel R2308GZ4GC (HPC 専用 2U ラックマウント型サーバ)
CPU	Intel Xeon E5-2680 v2 (2.8GHz) ×2CPU(計 20 コア)/ノード
コプロセッサ	Intel Xeon Phi 5110P ×1 基(計 60 コア)/ノード
メモリ	64 GB/ノード

・GPU 搭載共用フロントエンド

ハードウェア	uniV UNI-i7GH
CPU	Intel Core i7 7700K (4.2GHz) ×1CPU(計 4 コア)/ノード
GPU	NVIDIA Quadro P1000 4GB ×1/ノード
メモリ	64 GB/ノード

・vPU 搭載共用フロントエンド

ハードウェア	NEC SX-Aurora TSUBASA A100-1
CPU	Intel Xeon Silver 4108 (1.80GHz) ×1CPU(計 8 コア)/ノード
ベクトルエンジン	SX-Aurora TSUBASA Type 10C ×1/ノード
メモリ	96 GB/ノード

・インターネット高速転送サーバ

ハードウェアをログインサーバと共用します。

- ・ログインサーバ

2 ノードによる構成です。インターネット高速転送サーバとハードウェアを共用します。

1.3. ソフトウェア構成

各計算機に導入するソフトウェアの構成を示します。

1.3.1. オペレーティングシステム (os)

各システムで採用するオペレーティングシステム (os) を示します。

表 1.3.1 オペレーティングシステム (os)

システム名	オペレーティングシステム (os)
ログインサーバ	CentOS 6.6 (x86_64)
共用フロントエンドサーバ	Red Hat Enterprise Linux 6.6 (x86_64)
GPU 搭載フロントエンド	CentOS 6.9 (x86_64)
VPU 搭載フロントエンド	CentOS 7.3 (x86_64)
演算ノード (A,B,D,G システム)	CentOS 6.6 (x86_64)
演算ノード (F,H システム)	CentOS 6.8 (x86_64)
演算ノード (V システム)	CentOS 7.3 (x86_64)
演算ノード (W システム)	CentOS 7.7 (x86_64)
演算ノード (X システム)	CentOS 8.1 (aarch64)
HPCI ブリポスト	CentOS 7.3 (x86_64)

1.3.2. ソフトウェア

FOCUS スパコンシステムで利用可能な主なソフトウェアの一覧を示します。

表 1.3.2-1 ソフトウェア一覧

	フロントエンドサーバ	演算ノード
ジョブスケジューラ		
Slurm Workload Manage 17.02.10	○	○
開発環境 (コンパイラ)		
インテル® Parallel Studio XE 2016 Cluster Edition Update 2	○	○
インテル® Parallel Studio XE 2017 Cluster Edition Update 1	○	○
インテル® Parallel Studio XE 2017 Cluster Edition Update 6	○	○
インテル® Parallel Studio XE 2018 Cluster Edition Update 3	○	○
GNU 4.4.7	○	○
GNU 6.3.0	○	○
ソフトウェア		
Java SDK 1.8.0_172	○	○
Emacs 23.1.1	○	—
vim 7.2	○	—
OpenMPI 1.10.7	○	○
OpenMPI 2.1.1	○	○
OpenMPI 2.1.3	○	○
OpenMX 3.7.6	○	○
OpenMX 3.8.1	○	○
GAMESS May 1,2013	○	○
GAMESS Aug 18,2016	○	○
ABINIT-MP 4.1	○	○
ABINIT-MP 6.0	○	○
ABINIT-MP 7.0	○	○
NAMD 2.9	○	○
GROMACS 4.6.5	○	○
GROMACS 2018.4	○	○
LAMMPS 28Jun14	○	○
LAMMPS 30Jul16	○	○
Quantum ESPRESSO 5.0.2	○	○
Quantum ESPRESSO 5.2.1	○	○
Quantum ESPRESSO 6.1	○	○
OpenFOAM 2.4.x	○	○
OpenFOAM 3.0.0	○	○
OpenFOAM 4.0	○	○
OpenFOAM v1806	○	○
ParaView 4.0.1	○	○
gnuplot 5.2.6	○	○
Octave 3.6.4	○	○
Pov-Ray 3.6.1	○	○
R 3.3.2	○	○
GLview 20131211	○	○
AutoDock Vina 1.1.2	○	○
NTChem 2013.9.0	○	○

GNU Scientific Library 2.5	○	○
cmake 3.12.1	○	○
Python 2.7.12	○	○
Python 3.5.2	○	○
FDPS 1.1	○	○

1.3.3. アプリケーション

/home1/share/にインストールしたアプリケーション、数値計算ライブラリ等のご自由に利用ください。FOCUS スパコン上で動作検証済みの商用・有償アプリケーション（Gaussian、MIZUHO/BioStation 以外）は、ソフトウェアベンダーからライセンスを取得して頂きまして利用頂けます。

その他フリーソフト等、各利用者がホームディレクトリ配下に独自にインストールしたものが利用可能です。

1.3.3.1. ライセンスが必要なアプリケーション(商用・有償)

a) コンパイラ

Intel コンパイラは、導入済みのライセンスがありますので、使用していただくことができます。

（ただしまれに同時にご使用の方が多いたタイミングで、導入しておりますライセンス数を越えてしまい、ご使用のためのライセンスがありませんというエラーが出るがありますが、そのような場合はしばらく時間を置いてご利用下さい。）

b) Gaussian、MIZUHO/BioStation

Gaussian や MIZUHO/BioStation を FOCUS スパコンで利用する場合は、ライセンスを準備する必要はありませんが、演算ノードの利用料とは別にノード時間に応じて利用料がかかります。これらアプリケーションの利用を希望するユーザは、OKBiz (<https://secure.okbiz.okwave.jp/focus/>) より利用希望する旨連絡してください。

c) 上記以外の商用・有償ソフトウェア

上記以外の商用・有償ソフトウェアを FOCUS スパコンで利用する場合は、各アプリケーションのベンダーに問い合わせして下さい。（FOCUS スパコンに未導入の商用ソフトウェアを利用されたい場合は、各ソフトウェアベンダーに FOCUS スパコンで使いたいとご相談下さい。）

1.3.3.2. フリーソフトウェア

/home1/share にインストールされているアプリケーション、ライブラリ等は自由に使用していただくことができます。しかしながら当該アプリケーションのご利用にあたっては FOCUS によるサポートはありません。

利用者あるいはグループとして FOCUS スパコンに未導入のフリーソフトウェアを利用されたい場合は、各アカウントのホームディレクトリやグループ共有のホーム領域 (/home1/グループ名/share) に自由にインストールして利用して頂いてかまいません。基本的に FOCUS からのサポートはありません。

1.3.3.3. アプリケーションに関するサポート

FOCUS スパコンに導入した商用ソフトウェアのサポートは、各ソフトウェアベンダーが行っています。

提供されているサポートサービスについては、以下をご参照ください。

公益財団法人 計算科学振興財団 > FOCUS スパコン > 利用サポート

「アプリケーションに関するサポート」

<http://www.j-focus.or.jp/focus/support.html#h17285>

1.4. 製品マニュアル

SSL-VPN 接続でセンター内にある製品マニュアルを閲覧できます。下記の URL からアクセスしてご覧ください。

[URL] <https://portal.j-focus.jp/focus/app/>

表 1.4 マニュアル一覧

関連製品名	マニュアル名称	形式	和	英
インテル® Paralle Studio XE 2016	リリースノート	PDF	○	
	インテル Parallel Studio XE 2016 入門ガイド	PDF	○	
インテル® Paralle Studio XE Cluster Edition 関連ドキュメント	インテル CilkPlus ユーザーズガイド	PDF	○	
	コンパイラー OpenMP 入門	PDF	○	
	OpenMP 3.0 C/C++構文の概要	PDF	○	
	インテル OpenMP 互換ライブラリー 利用ガイド	PDF	○	
	コンパイラー最適化 クイック・リファレンス・ガイド	PDF	○	
インテル® Parallel Studion XE 2016 Composer Edition for C++ Linux	インテル コンパイラーリリースノート	PDF	○	
	インテル C++ コンパイラーのベクトル化ガイド	PDF	○	
	インテル C/C++ コンパイラー OpenMP 活用ガイド	PDF	○	
インテル® Parallel Studion XE 2016 Composer Edition for Fortran Linux	インテル Fortran ライブラリー・リファレンス	PDF	○	
	インテル Fortran コンパイラー OpenMP 活用ガイド	PDF	○	
インテル® MPI ライブラリー	リファレンス・マニュアル	PDF	○	
	ユーザーズガイド	PDF	○	
	リリースノート(英語)	PDF		○
インテル® Trace Analyzer & Collector	チュートリアル: MPI Performance Snapshot で MPI アプリケーションを解析する	PDF	○	
	チュートリアル: MPI アプリケーションの解析	PDF	○	
	リリースノート	PDF	○	
インテル® MKL	ユーザーズガイド	HTML	○	
	インテル MKL クックブック	PDF	○	
	チュートリアル (C++)	HTML	○	
	チュートリアル (Fortran)	HTML	○	
インテル® IPP	ユーザーズガイド	HTML	○	
	チュートリアル	HTML	○	
インテル® TBB	リリースノート(英語)	TXT		○
	ユーザーガイド & リファレンス・マニュアル	HTML	○	
	チュートリアル	HTML	○	
インテル® DAAL	ゲッティング・スタート	HTML	○	
	プログラミング・ガイド	HTML	○	
インテル® VTune Amplifier XE	パフォーマンス解析入門ガイド	PDF	○	
	リリースノート(英語)	PDF		○
	チュートリアル	HTML		○
インテル® Inspector XE	リリースノート(英語)	PDF		○
SLURM 15.08	Documentation	HTML		○
MIZUHO / ABNIT-MP	利用マニュアル	PDF	○	

1.5. ディレクトリ構成

ご利用の形態やソフトウェア、ジョブの特性によってディレクトリ（ストレージシステム）を使い分けていただくようお願い致します。

1.5.1. 共用フロントエンドサーバ

共用フロントエンドサーバのディレクトリ構成は下表のとおりです。ログインサーバは同じストレージ領域をマウントしております。

表 1.5.1 ディレクトリ構成（共用フロントエンドサーバ）

ディレクトリパス	利用目的
/home1/グループ名/アカウント名	ホームディレクトリ
/home1/グループ名	グループ共有のホーム領域。容量は課題（グループ）あたり 200GB
/home1/グループ名/share	グループで共用するソフトウェアを格納するための領域
/home3/グループ名	グループ共有の追加ストレージ領域（フロントエンドサーバのみ）
/home1/グループ名/アカウント名/skeed または /home3/グループ名/アカウント名/skeed	インターネット高速転送システムでデータ授受を行う場合に使用。 専用ソフトウェアでは"/"と表示される。
/home1/share	システム全体で共用するソフトウェアを格納（フロントエンドサーバのみ）

※グループ名は「g」＋「課題 ID」、アカウント名は「u」＋「課題名」＋数字 4 桁です。

（参考）

/home1	NAS ストレージシステム	（書き込み性能 全体で 500MB/s）
/home3	分散ファイルシステム	（書き込み性能 全体で 11GB/s、1 プロセスあたり最大 1GB/s）

1.5.2. 演算ノード

演算ノードのディレクトリ構成は下表のとおりです。

表 1.5.2 ディレクトリ構成（演算ノード）

ディレクトリパス	利用目的
/work	スクラッチディレクトリ（演算ノードのローカルディスク）

※演算ノードにログインすることはできません。

（参考）/work の書き込み性能は各システムにより異なります。

実測値は「付録 A. FOCUS スパコンシステム各種サーバ・ストレージ概要」をご参照ください。

2. システムの利用方法

FOCUS スパコンシステムの利用方法について以下に示します。

2.1. システムへのログイン

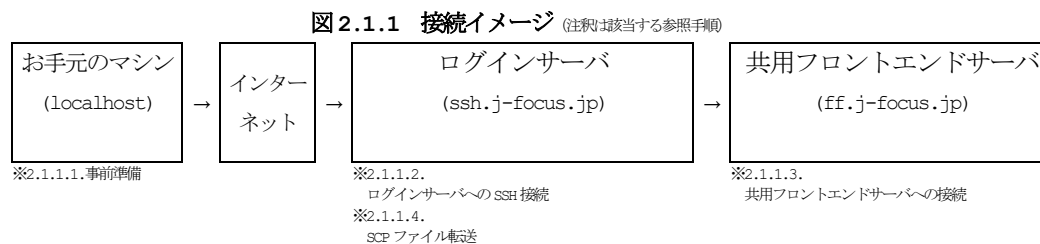
インターネットから、FOCUS スパコンシステムにログインするには、SSH 鍵交換による接続法と SSL-VPN による接続法の2経路があります。接続サーバ名等は、「付録 A. FOCUS スパコンシステム各種サーバ・ストレージ概要」をご参照ください。

以下では、ログインの詳細な方法を記述します。

2.1.1. インターネットからの SSH 接続によるログイン

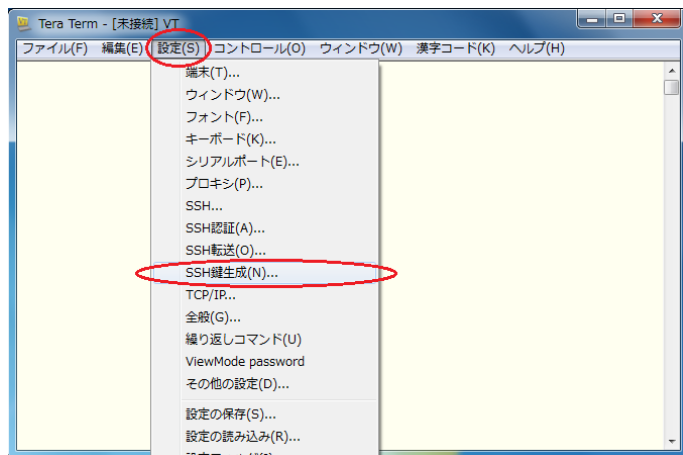
お手持のマシンがインターネットに対して SSH 接続できる環境であれば、SSH 接続によりセンター内のシステムにアクセスすることができます。

以下の『2.1.1.1 事前準備（秘密鍵・公開鍵の作成と登録）』を実施して、公開鍵の登録が完了した後、ログインサーバ経由で共用フロントエンドサーバに接続できます。



2.1.1.1. 事前準備（秘密鍵・公開鍵の作成と登録）

- (1) スタートメニュー [すべてのプログラム] → [Tera Term] を順に選択します。
- (2) Tera Term『新しい接続』画面で [キャンセル] ボタンをクリックします。
(『新しい接続』画面を閉じます。)
- (3) Tera Term メニュー [設定] → [SSH 鍵生成] を順に選択します。



- (4) 『TTSSH: 鍵生成』画面で、鍵の種類「RSA」(RSA2)を選択し、[生成] ボタンをクリックします。
(注)鍵の種類「RSA1、RSA、DSA」の中央に位置する「RSA」がRSA2 です。

- (5) 「鍵のパスフレーズ」と「パスフレーズの確認」に同じフレーズを入力し、[公開鍵の保存] ボタンと [秘密鍵の保存] ボタンを押し、公開鍵と秘密鍵を保存します。

- (6) OKBiz (<https://secure.okbiz.okwave.jp/focus/>) で公開鍵 (id_rsa.pub) の登録を依頼します。

※ファイル名の初期値は次のとおりです。

- 公開鍵: id_rsa.pub ←OKBiz に添付するのは“~.pub”の方です。
- 秘密鍵: id_rsa ←お手元で管理してください (添付しないでください)。

※OKBiz が使用不可の場合はメールで運用課 unyo@j-focus.or.jp に送付してください。

※秘密鍵は SSH 接続する際に使用します。

※秘密鍵はお手元で厳重に管理をお願いします (絶対に公開鍵と一緒に送付しないで下さい)。

2.1.1.2. ログインサーバへの SSH 接続

前述の手順『2.1.1.1 事前準備（秘密鍵・公開鍵の作成と登録）』で作成した秘密鍵を使って、インターネットからログインサーバに対して SSH 接続を行います。SSH 接続に関わる各種情報は次のとおりです。

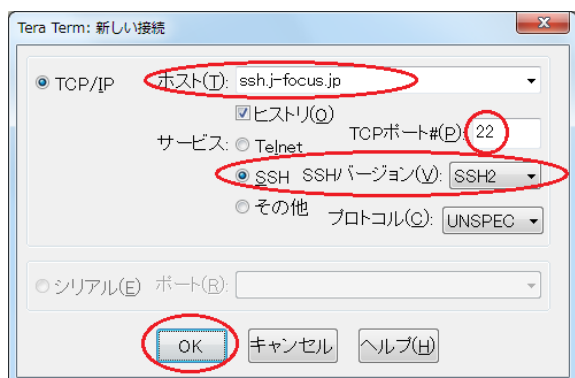
接続先	サービス	ポート番号	サービスバージョン	認証方式
ssh.j-focus.jp	SSH	22	SSH2	公開鍵認証

ご使用の環境によってはプロキシを設定する必要があります。そのような場合は、ご所属のネットワーク管理者にご確認ください。

手順は次のとおりです。

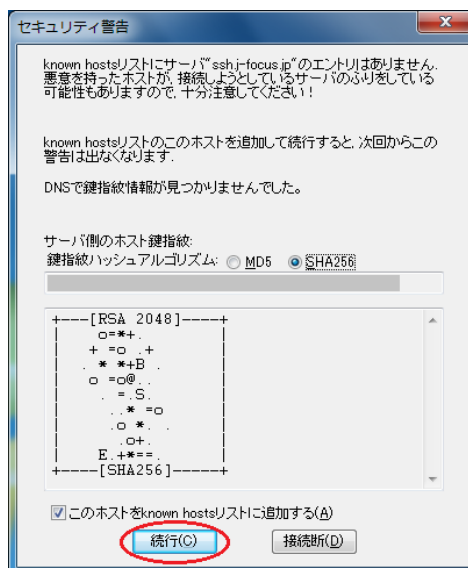
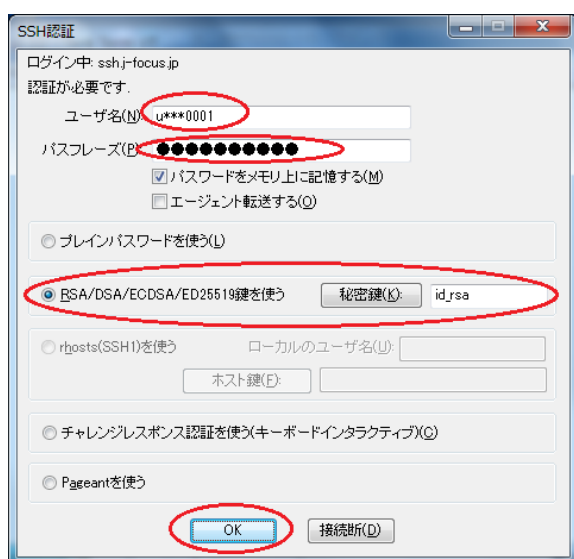
- (1) スタートメニュー [すべてのプログラム] → [Tera Term] を選択します。
- (2) TeraTerm『新しい接続』画面で以下の指定を行い、[OK] ボタンを押します。

- ・ホスト名 : ssh.j-focus.jp
- ・サービス : SSH
- ・TCP ポート# : 22
- ・SSH バージョン : SSH2

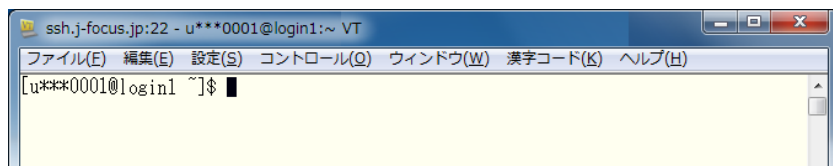


- (3) 『SSH 認証』画面で以下の指定を行い、[OK] ボタンを押します。
なお、セキュリティ警告ウィンドウが現れた場合は、[続行] ボタンを押します。

- ・ユーザ名 : センターから発行されたアカウント名（「u」+「課題名」+数字 4 桁）
- ・パスフレーズ : 公開鍵・秘密鍵を作成した際に指定したパスフレーズ
- ・RSA/DSA 鍵を使う : チェックを入れる
- ・秘密鍵 : お手元のマシンに保存している秘密鍵ファイル



- (4) プロンプト（[アカウント名@login1~]\$ もしくは[アカウント名@login2~]\$）が表示されることを確認します。



2.1.1.3. 共用フロントエンドサーバへの接続

ssh コマンドにより共用フロントエンドサーバ(ff01/ff02)は ff に接続します。
GPU 搭載共用フロントエンド(fgpu1)は、 fgpu1 に接続します。
VPU 搭載共用フロントエンド(fvpu1)は、 fvpu1 に接続します。

【共用フロントエンド利用についての注意点】

共用フロントエンド(ff01/ff02/fgpu1/fvpu1)上では、プログラムの開発、小規模な解析・デバッグ、小規模なプリポスト処理の実行が許可されます。
ユーザーは下記の範囲での実行が可能です。

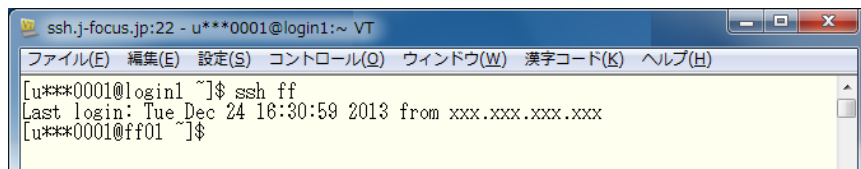
- CPU 時間 1 時間 (1 時間で強制終了となります)
- プロセス数 1 プロセス (並列実行、複数プロセスの起動は禁止です)
- 利用メモリ 1 GB 程度 (小規模処理のみ許可)

上記範囲を越える場合は、バッチ処理より演算ノード上でジョブを実行してください。
または、専用フロントエンド (有償) の利用をご検討ください。

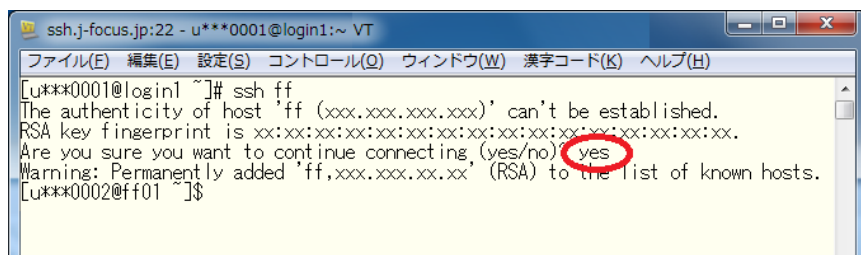
※専用フロントエンド上でのジョブ実行については、
実行時間、数、規模の制限はありません。(別途、申請書の提出が必要です。)

【ff への接続】

```
ssh ff
```



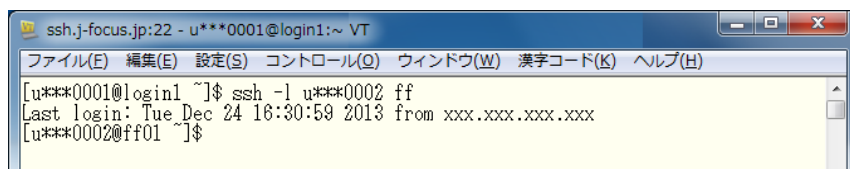
なお、初回接続時は確認メッセージが表示されるので、「yes」を入力します。



また、異なるアカウント名で接続する場合は、ssh コマンドの -l オプションを使ってアカウント名を指定します。

【ff への接続 (アカウント指定あり)】

```
ssh -l アカウント名 ff
```

**【fgpu1 への接続】**

前述の例の「ff」を「fgpu1」に読み替えて指定してください。

```
ssh fgpu1
```

【fvpu1 への接続】

前述の例の「ff」を「fvpu1」に読み替えて指定してください。

```
ssh fvpu1
```

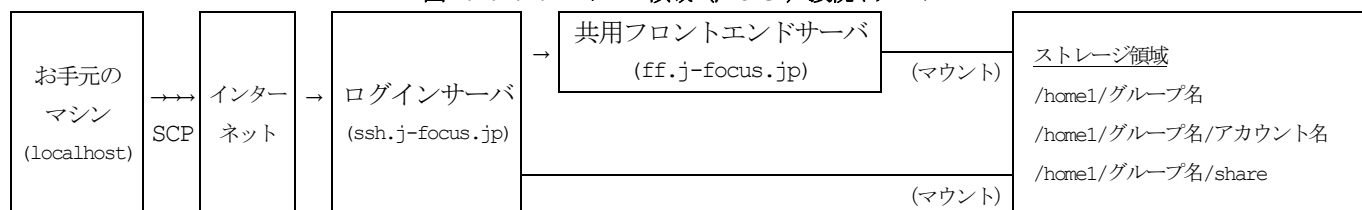
2.1.1.4. SCP ファイル転送

インターネットからログインサーバに対して、SCP ファイル転送を行います。事前準備として秘密鍵の変換操作が必要となりますので、次の①、②の順で操作を行います。

- ① 事前準備（秘密鍵の変換） ・ ・ ・ PuTTY Key Generator (PuTTYgen) 使用
- ② SCP ファイル転送 ・ ・ ・ WinSCP

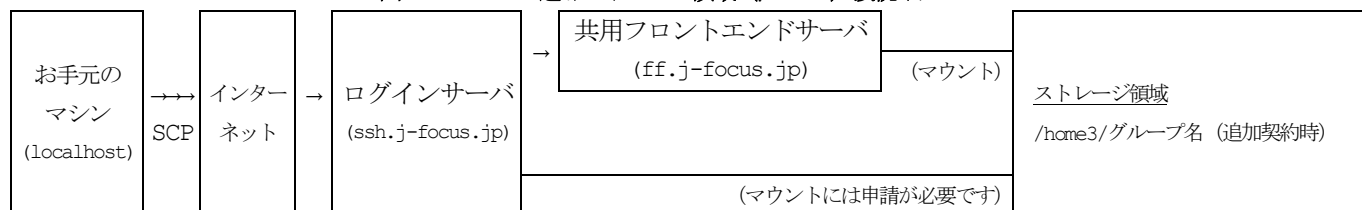
ホーム領域 (/home1) は、ログインサーバと共用フロントエンドサーバが同じファイルシステムをマウントしているため、ファイルの転送操作はログインサーバに対してのみ実施します。

図 2.1.1.4.1 ホーム領域 (/home1) 接続イメージ



追加ストレージ領域 (/home3) は、初期状態では共用フロントエンドサーバのみがマウントしています。ログインサーバからのマウントには、ユーザ単位での申請が必要となります。

図 2.1.1.4.2 追加ストレージ領域 (/home3) 接続イメージ

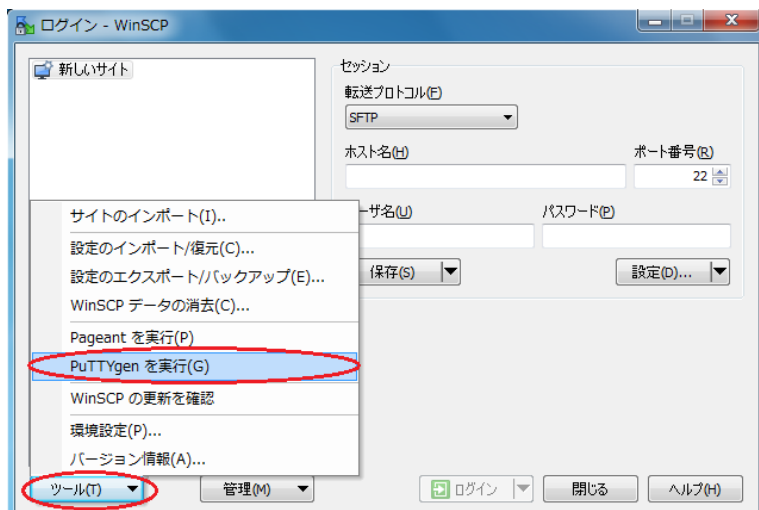


なお、ご使用の環境によってはプロキシを設定する必要があります。そのような場合は、ご所属のネットワーク管理者にご確認ください。

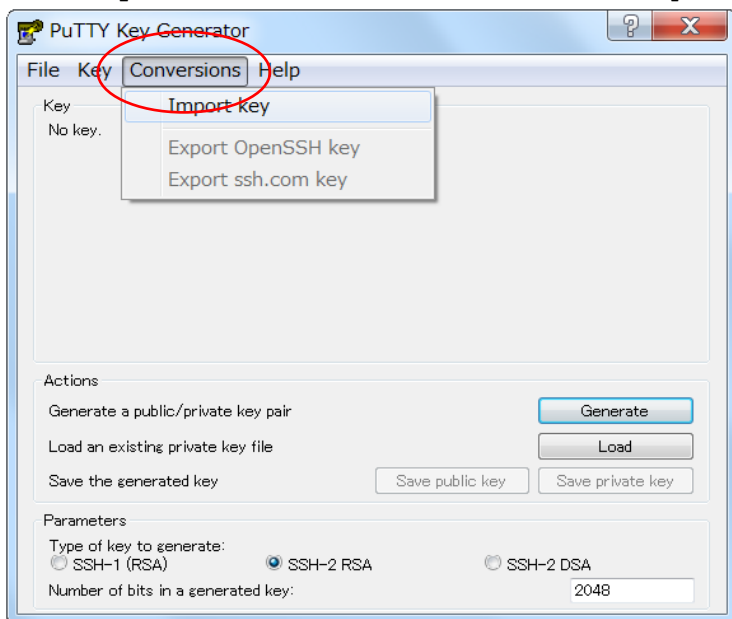
① 事前準備（秘密鍵の変換）

WinSCP 用に秘密鍵を PuTTY 形式に変換します。前述『2.1.1.1 事前準備（秘密鍵・公開鍵の作成と登録）』の手順で作成した秘密鍵（id_rsa）を使います。

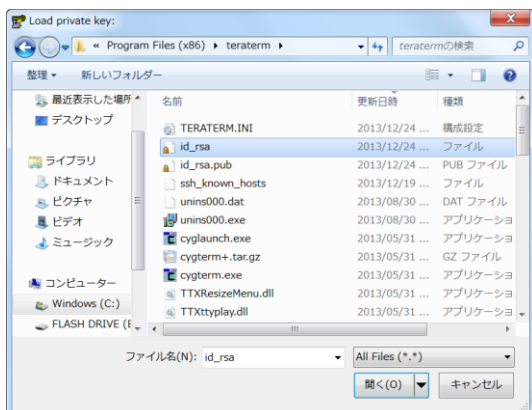
- (1) スタートメニュー [すべてのプログラム] → [WinSCP] → 『ログイン - WinSCP』画面で [ツール] → [PuTTYgen を実行] を順に選択します。
(PuTTY Key Generator を起動します。)



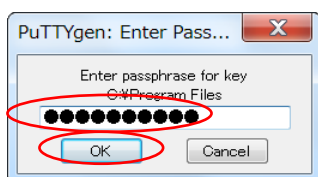
- (2) PuTTY Key Generator メニュー [Conversions] → [Import Key] を順に選択します。



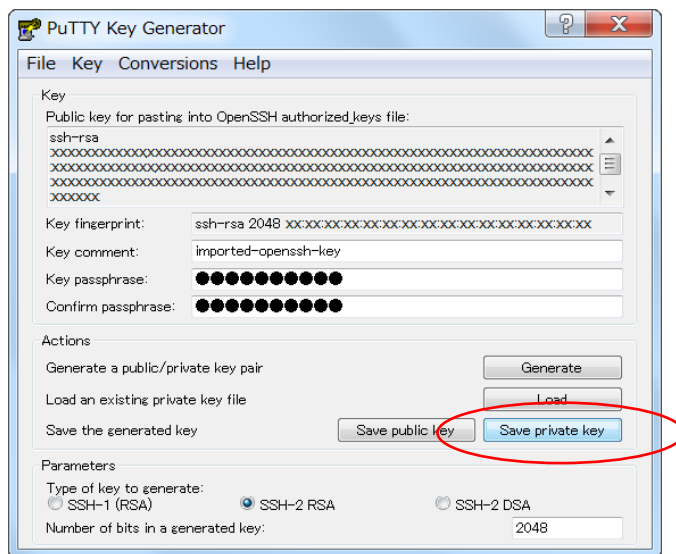
- (3) 前述『2.1.1.1 事前準備 (秘密鍵・公開鍵の作成と登録)』の手順で作成した秘密鍵を選択します。
- ※Windows 環境の画面例



- (4) 公開鍵・秘密鍵を作成した際のパスフレーズを入力し、[OK] ボタンをクリックします。



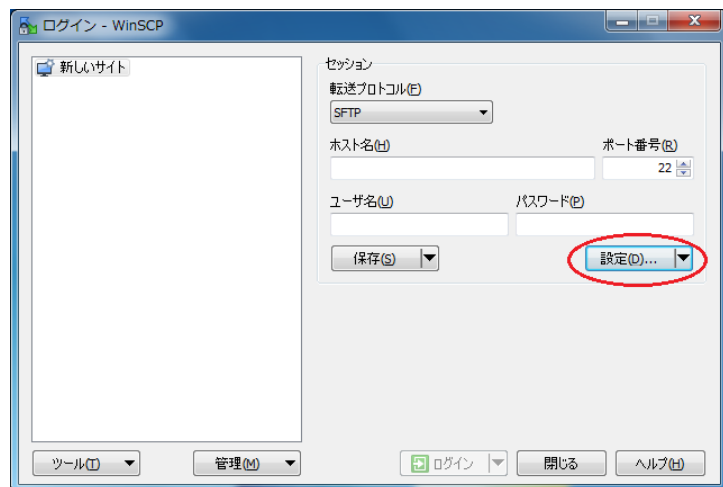
- (5) [Save private key] ボタンをクリックします (変換した鍵が保存されます)。



② SCP によるファイル転送 (WinSCP 使用)

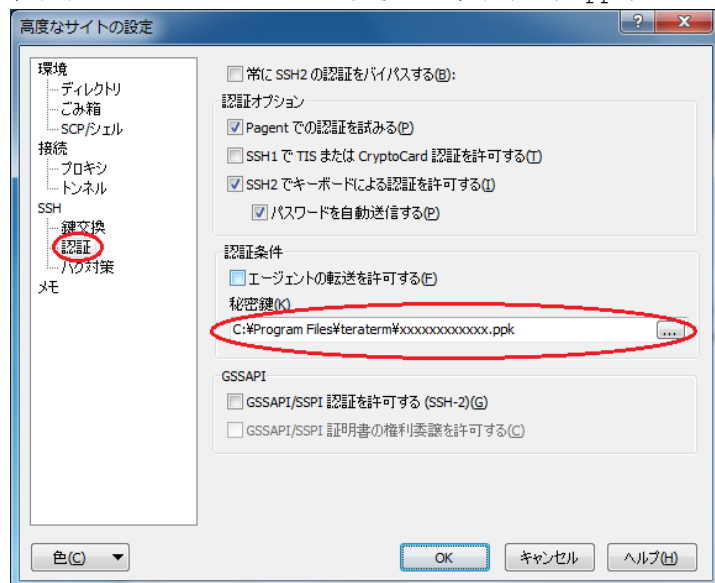
前述の手順『①事前準備 (秘密鍵の変換)』で PuTTY 形式に変換した秘密鍵を使用します。

- (1) スタートメニュー [すべてのプログラム] → [WinSCP] の順に選択します。
- (2) 『ログイン - WinSCP』画面で [設定] ボタンをクリックします。

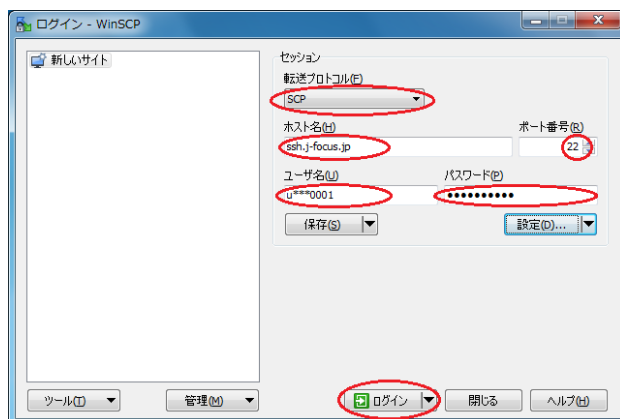


- (3) 『高度なサイトの設定』画面で [SSH] → [認証] を順に選択し、以下の指定を行ないます。

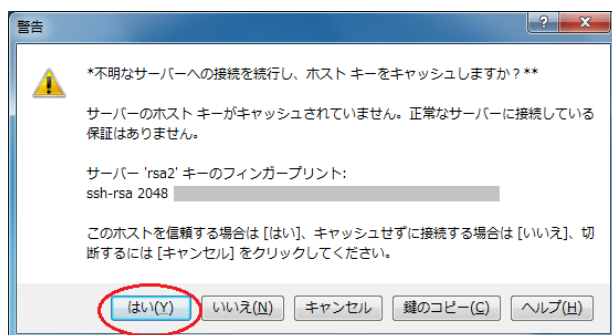
- ・秘密鍵 : PuTTY 形式に変換した秘密鍵 (~.ppk)



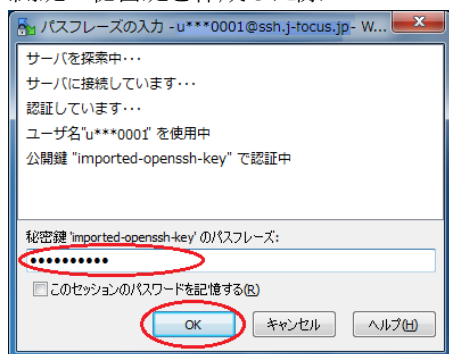
- (4) 『ログイン - WinSCP』画面に戻り以下の指定を行い、[ログイン] ボタンをクリックします。
- ・プロトコル : SFTP または SCP
 - ・ホスト名 : ssh.j-focus.jp
 - ・ポート番号 : 22
 - ・ユーザ名 : アカウント名 (「u」 + 「課題名」 + 数字 4 桁)
 - ・パスワード : 公開鍵・秘密鍵を作成した際のパスフレーズ



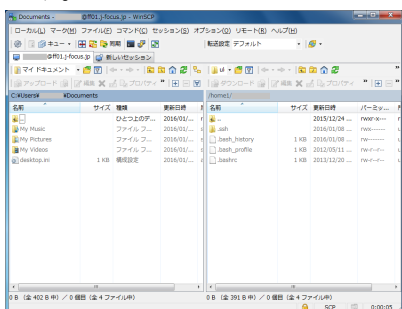
- (5) 『警告』画面が表示された場合は内容を確認し、[はい] ボタンをクリックします。



- (6) 公開鍵・秘密鍵を作成した際のパスフレーズを入力し、[OK] ボタンをクリックします。

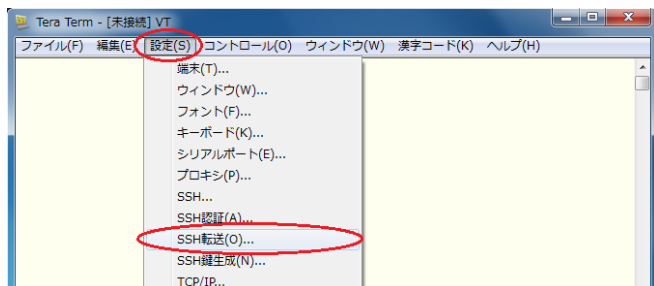


- (7) 以下のような画面が表示され、お手元のマシンとログインサーバ間でファイル転送が出来るようになります。

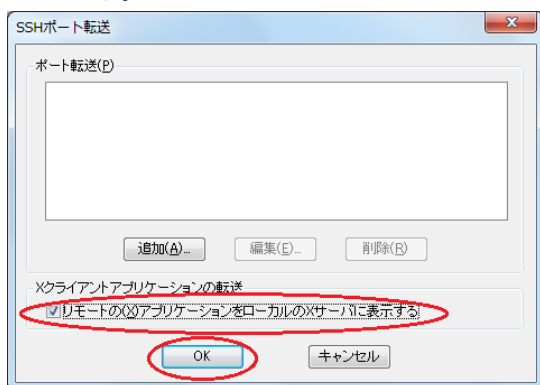


2.1.1.5. X Window System を利用するアプリケーションの使用方法 (Windows 環境)

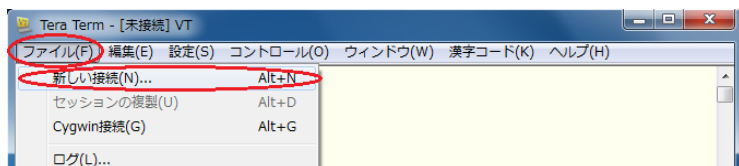
- (1) お手元のマシン (localhost) に Cygwin/X、Xming 等の X サーバソフトウェアをあらかじめインストールしておき、お手元のマシン (localhost) で X サーバを起動します。
- (2) スタートメニュー [すべてのプログラム] → [Tera Term] を順に選択します。
- (3) Tera Term『新しい接続』画面で [キャンセル] ボタンをクリックします。(『新しい接続』画面を閉じます。)
- (4) Tera Term メニュー [設定] → [SSH 転送] を選択します。



- (5) [リモートの(X)アプリケーションをローカル(X)サーバに表示する] にチェックを入れ、[OK] をクリックします。

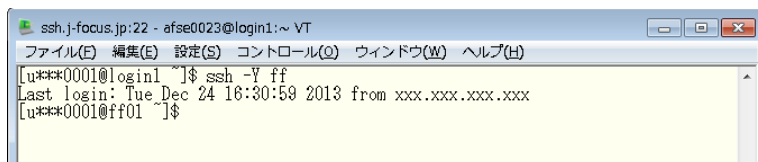


- (6) Tera Term メニュー [ファイル] → [新しい接続] を順にクリックします。(『新しい接続』画面を開きます。)



- (7) ログインサーバに SSH 接続します。(参照：『2.1.1.2 ログインサーバへの SSH 接続』)
- (8) ログインサーバから共用フロントエンドサーバ (ff01 または ff02) に ssh 接続する際に、-Y (大文字ワイ) オプションを付けます。

```
ssh -Y ff
```



- (9) 以上で、X Window System を利用するアプリケーションをお手元のマシンで利用出来るようになります。

2.1.1.6. インターネットからの SSH 多段接続によるログイン方法

前述 2.1.1 ではお手元のマシン (localhost) からログインサーバに接続し、そこから共用フロントエンドサーバに接続する方法を示しました。本節では、SSH 多段接続により localhost から直接フロントエンドサーバにログインする方法について示します。

・proxy 設定による SSH 多段接続の設定方法

cygwin (Windows 環境) , MacOS X, Linux 等の localhost で、\$HOME/.ssh/config ファイルを以下のように設定します。ログインする共用フロントエンドサーバは ff01.j-focus.jp、アカウント名は user0001 の場合の設定例になります。

【\$HOME/.ssh/config 設定例】

```
Host FocusLogin                                #
    HostName ssh.j-focus.jp                    # フロントエンドサーバの設定
    User user0001                              # アカウント名
    Port 22                                    # ポート番号
    IdentityFile ~/.ssh/id_rsa                 # 秘密鍵の保管場所

Host ff01Focus
    HostName ff01.j-focus.jp                    # ホストの指定
    User user0001                              #
    ProxyCommand ssh FocusLogin nc %h %p        #
```

config ファイルでの設定終了後、フロントエンドへの接続を実行します。

【多段 SSH 接続実行例】

```
[localhost] $ ssh ff01Focus
Enter passphrase for key '~/.ssh/id_rsa':      ←公開鍵・秘密鍵のパスフレーズ
user0001@ff01's password:                      ←サーバログイン用のアカウントパスワードを入力
Last login: Wed Aug 29 14:05:10 2013 from login2.j-focus.jp
[user0001@ff01 ~]$ hostname
ff01.p
[user0001@ff01 ~]$
```

コマンドを使って localhost から直接フロントエンドサーバへファイルのコピーを行うことも可能です。

【多段 SSH 接続を使った scp 実行例】

```
[local host] $ scp testfile ff01Focus:~        ←testfile を ff01 のホームにコピー
Enter passphrase for key '~/.ssh/id_rsa':      ←公開鍵・秘密鍵のパスフレーズ
user0001@ff01's password:                      ←サーバログイン用のアカウントパスワードを入力
testfile                                         100% 217      0.2KB/s   00:00
[localhost] $
```

・ Tera Term マクロ を使った SSH 多段接続の設定方法

Tera Term マクロを使い、ログインサーバへの SSH 接続を自動化します。

- (1) テキストエディタを使ってマクロを作成します。
以下では「FOCUS_FF.ttl」という名前でファイルを作成しています。
(接尾語は「.ttl」にしてください。)

```
LOGPATH  = 'C:¥<ログを保存したいパス>'
HOSTNAME1 = 'ssh.j-focus.jp'
HOSTNAME2 = 'ff.j-focus.jp'
USERNAME = 'アカウント名'
KEYFILE  = 'C:¥<事前準備で作成した秘密鍵の保管場所>¥id_rsa'

COMMAND = HOSTNAME1
strconcat COMMAND ':22 /ssh /auth=publickey /user='
strconcat COMMAND USERNAME
strconcat COMMAND ' /keyfile='
strconcat COMMAND KEYFILE
strconcat COMMAND ' /ask4passwd'
connect COMMAND

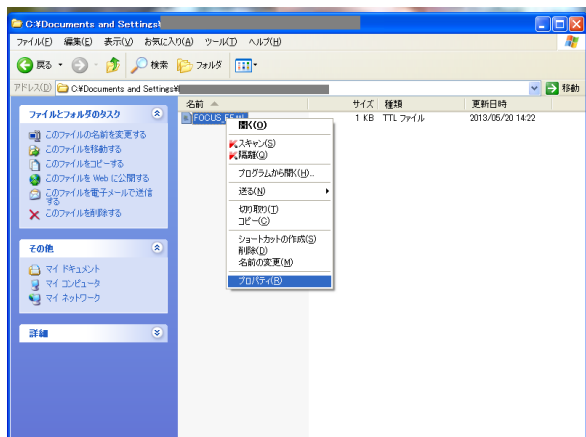
LOGFILE = LOGPATH
strconcat LOGFILE HOSTNAME1
getdate datestr "%Y%m%d-%H%M%S"
strconcat LOGFILE '-'
strconcat LOGFILE datestr
strconcat LOGFILE '.log'

logopen LOGFILE 0 0 1 1 1

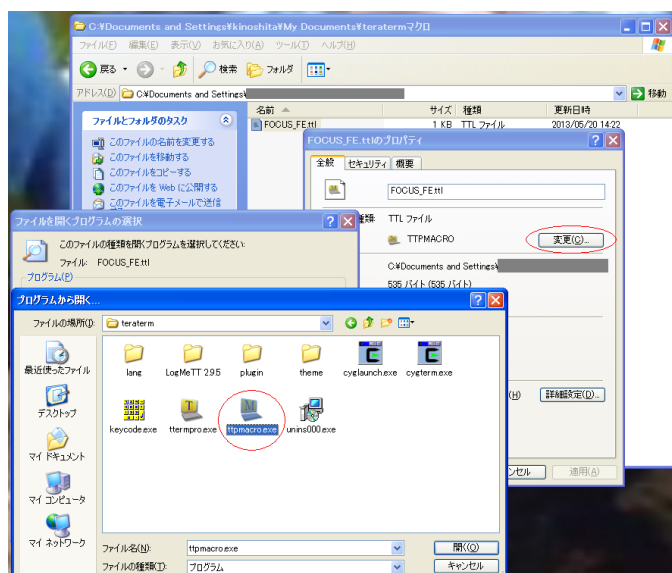
remote_prompt = '$'
wait remote_prompt

COMMAND = 'ssh '
strconcat COMMAND username
strconcat COMMAND '@'
strconcat COMMAND hostname2
sendln COMMAND
```

- (2) エクスプローラを開き、該当ファイル（例「FOCUS_FF.ttl」）を右クリックし「プロパティ」を選択します。

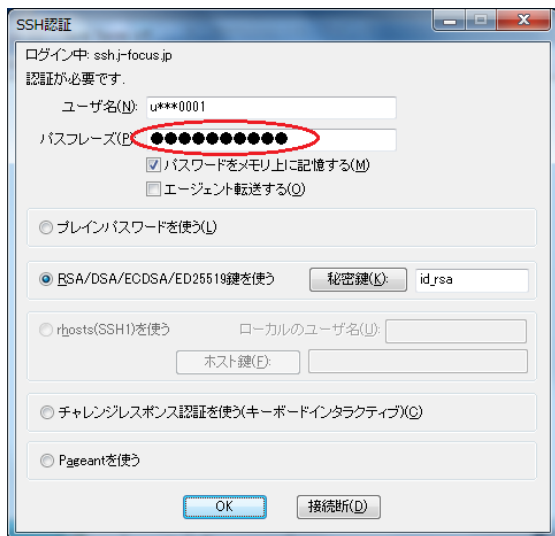


- (3) 「ファイルの種類」→「変更」→「このファイルの種類を開くプログラムを選択」→「httpmacro.exe」を選択します。

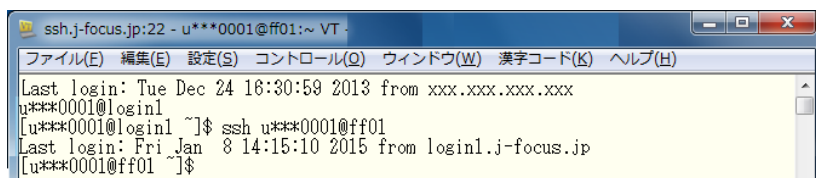


- (4) 接尾語「.ttl」をもつファイル（例「FOCUS_FF.ttl」）をダブルクリックすると Tera Term が起動します。

- (5) 「SSH 認証」画面でパスワードを入力し、[OK] ボタンをクリックします。



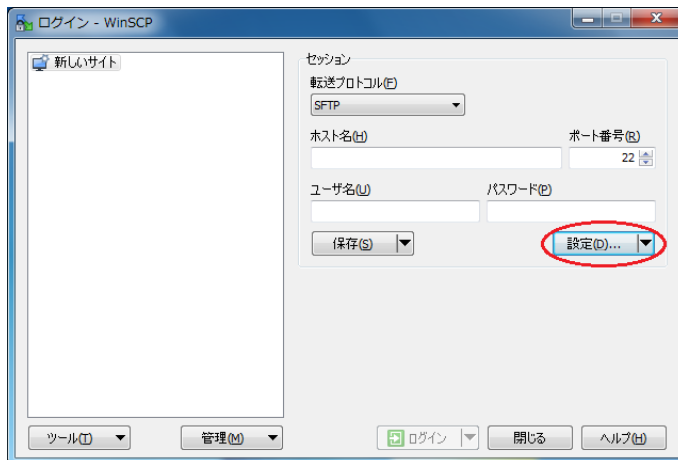
- (6) 以上の手順でフロントエンドサーバにログインできます。(「アカウント名@ff」 コマンドは自動的に入力されます。)



・SCP によるファイル転送（インターネットからの SSH 多段接続）

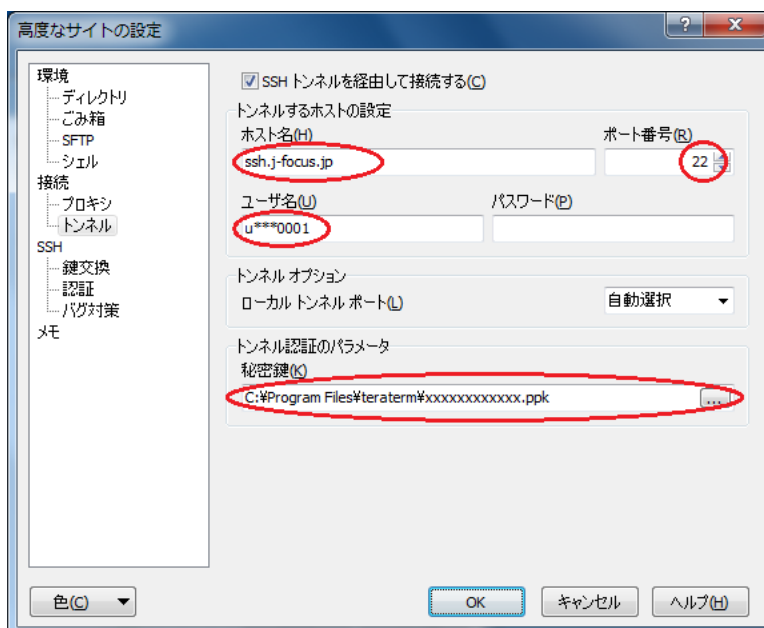
ここでは前述の『2.1.1.4 SCP ファイル転送』①事前準備で変換した秘密鍵を使用します。

- (1) スタートメニュー [すべてのプログラム] → [WinSCP] を順に選択します。
- (2) 『ログイン - WinSCP』画面で「設定」ボタンをクリックします。

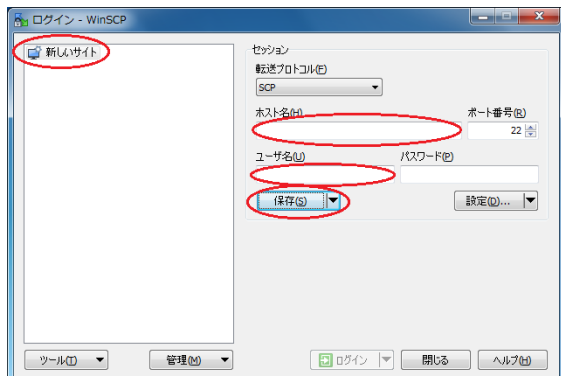


- (3) [接続] → [トンネル] を順に選択し、以下の指定を行います。

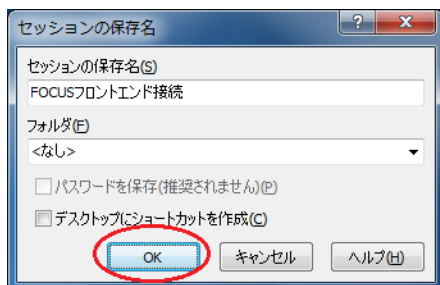
- ・SSH トンネルを経由して接続 : チェックを入れる
- ・ホスト名 : ssh.j-focus.jp
- ・ポート番号 : 22
- ・ユーザ名 : アカウント名（「u」＋「課題名」＋数字 4 桁）
- ・秘密鍵 : WinSCP 用に変換した PuTTY 形式の秘密鍵を指定
(鍵の変換方法は 2.1.1.4 SCP ファイル転送の①を参照)



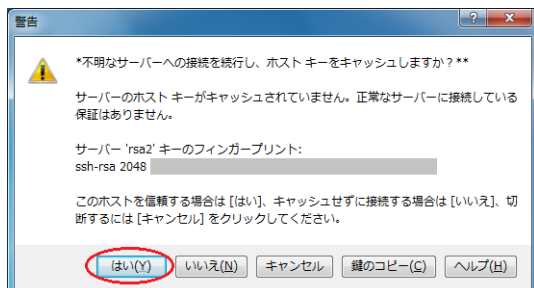
- (4) 「新しいサイト」を選択し、以下を指定し、[保存] ボタンをクリックします。
- ・ホスト名: ff01 または ff02
 - ・ユーザ名: アカウント名 (「u」+「課題名」+数字4桁)



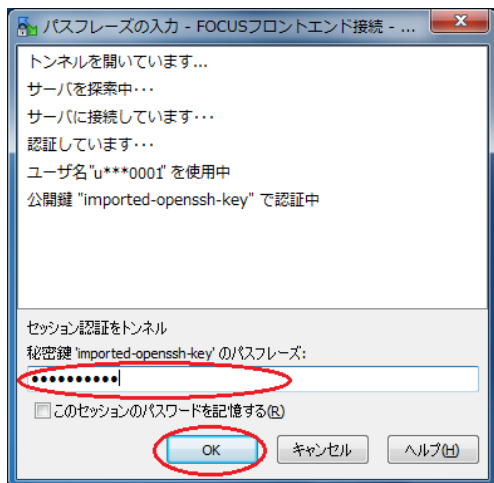
- (5) 『セッションの保存名』画面で保存名（例：FOCUS フロントエンド接続）を入力し、[OK] ボタンをクリックします。



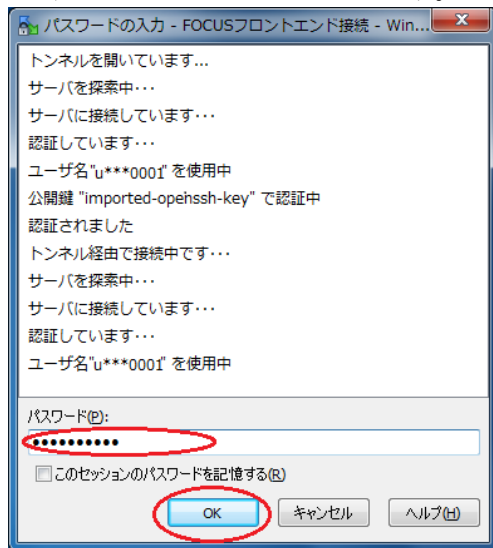
- (6) 『警告』画面（初回アクセス時のみ表示されます）が表示されたら、内容を確認したうえで、[はい] ボタンをクリックします。



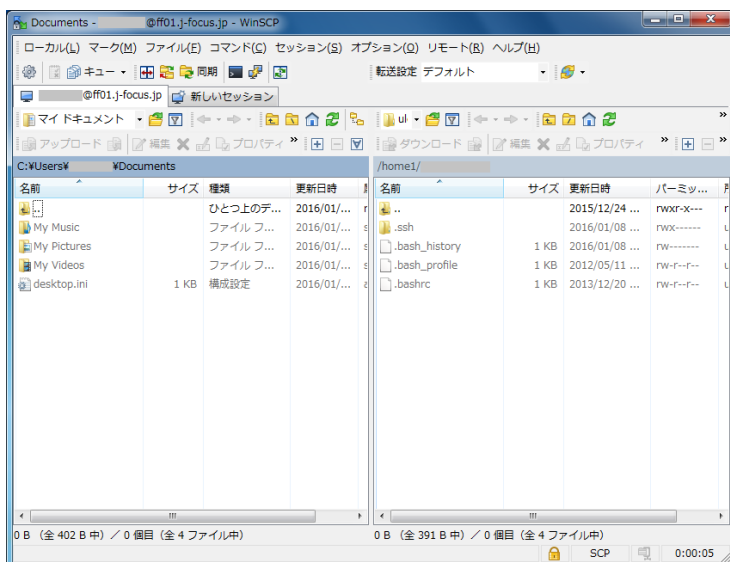
- (7) 『パズフレーズの入力』画面で公開鍵・秘密鍵のパズフレーズを入力し、[OK] ボタンをクリックします。



(8) アカウントのパスワードを入力します。



(9) 以上でお手元のマシン (localhost) とフロントエンドサーバ間でファイル転送が出来るようになります。



2.1.2. インターネットからの SSL-VPN 接続による利用方法

お手元のマシンがインターネットに対して SSL-VPN 接続できる環境であれば、SSL-VPN 接続でアクセスすることができます。また、ログインするときのホスト名、接続プロトコルは下表のとおりです。

表 2.1.2 ホスト名/接続プロトコル

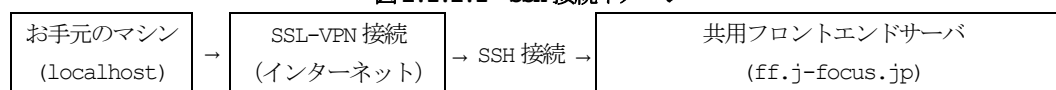
システム名	ホスト名	接続プロトコル			
		SSH	telnet	SCP	FTP
共用フロントエンドサーバ (共用利用向け)	ff01, ff02 fgpu1, fvpul1	○	X	○	X
専用フロントエンドサーバ (占有利用向け)	ft01, ..., ft04, fm01, ..., fm08, ff03, ff04				

共用フロントエンドサーバの利用については
「2.1.1.3.共用フロントエンドサーバへの接続」に記載の
【共用フロントエンド利用についての注意点】をご確認ください。

2.1.2.1. SSL-VPN 接続によるログイン

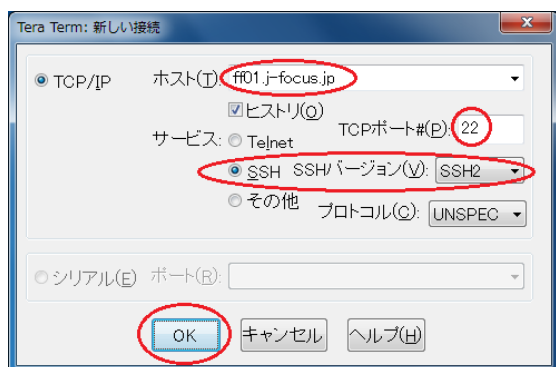
SSL-VPN 接続を使ってログインすることができます。

図 2.1.2.1 SSH 接続イメージ



手順は次のとおりです。

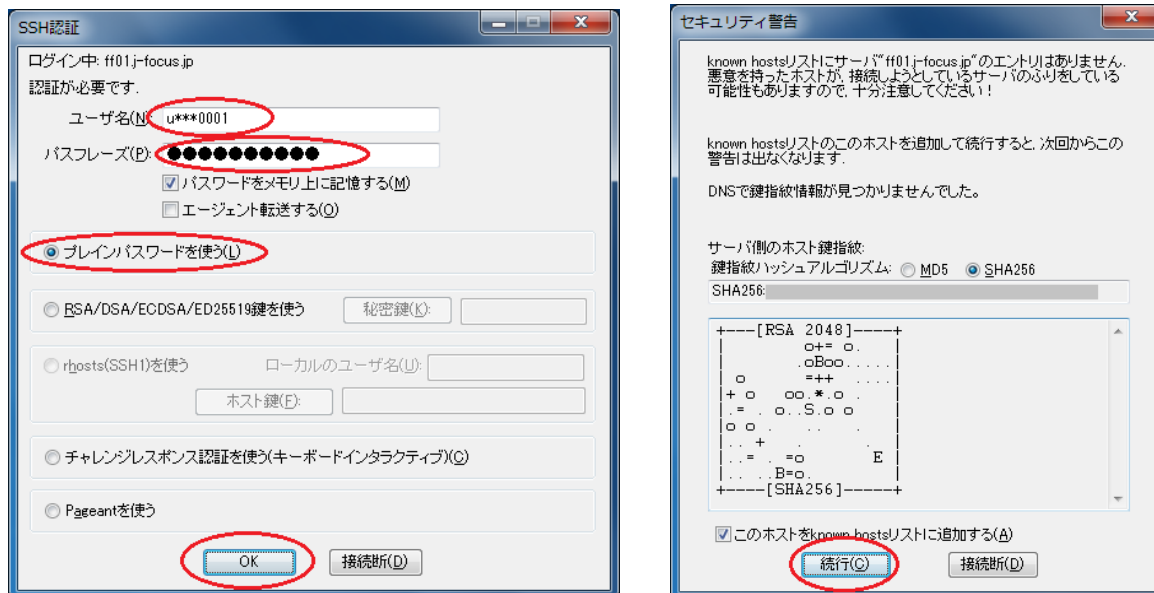
- SSL-VPN 接続を開始します。接続方法は『SSL-VPN 利用者マニュアル』(<http://www.j-focus.jp/sslvpn/>)を参照します。
- スタートメニュー [すべてのプログラム] → [Tera Term] → [Tera Term] を順に選択します。(Tera Term を起動します。)
- Tera Term『新しい接続』画面で以下を指定し、[OK] ボタンをクリックします。
 - ホスト名 : ff.j-focus.jp
 - サービス : SSH
 - SSH バージョン : SSH2



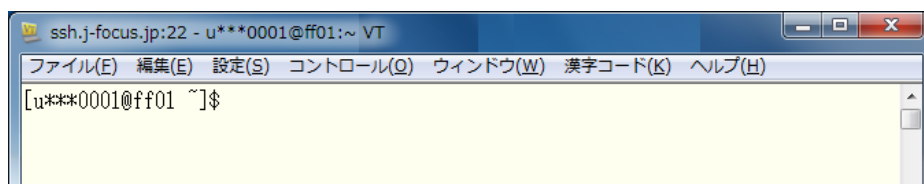
(4) 『SSH 認証』画面で以下を指定し、[OK] ボタンをクリックします。

- ・ユーザ名 : アカウント名 (「u」 + 「課題名」 + 数字 4 桁)
- ・パスフレーズ : パスワード
- ・プレインパスワードを使う : チェックする

操作の途中で『セキュリティ警告』画面が表示された場合は[続行] ボタンをクリックして手順を続けます。



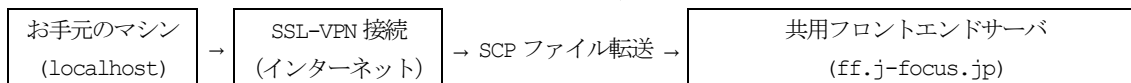
(5) 以上の手順で共用フロントエンドサーバへの ssh 接続に成功すると、以下のような画面が表示されます。



2.1.2.2. SSL-VPN 接続による SCP ファイル転送 (WinSCP 使用)

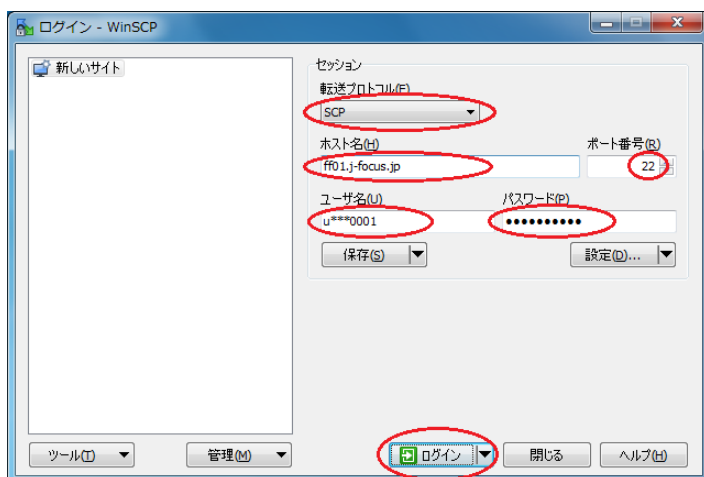
SSL-VPN 接続を使って、システムに SCP によるファイル転送を行います。

図 2.1.2.2 SCP ファイル転送

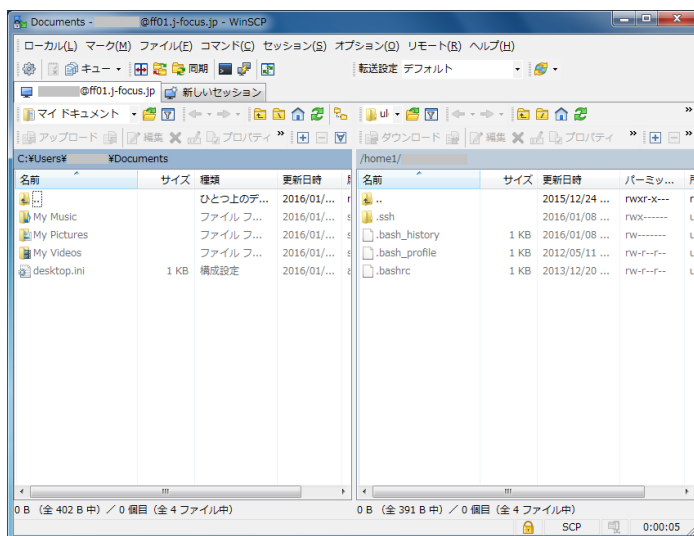


手順は次のとおりです。

- (1) SSL-VPN 接続を開始します。接続方法は『SSL-VPN 利用の手引き』 (<http://www.j-focus.jp/sslvpn/>) を参照します。
- (2) スタートメニュー [すべてのプログラム] → [WinSCP] を順に選択します。
- (3) 『WinSCP ログイン』画面で以下の指定を行い、[ログイン] ボタンをクリックします。
 - ・ファイルプロトコル : SFTP または SCP
 - ・ホスト名 : ff.j-focus.jp
 - ・ポート番号 : 22
 - ・ユーザ名 : アカウント名 (「u」 + 「課題名」 + 数字 4 桁)
 - ・パスワード : アカウントのパスワード



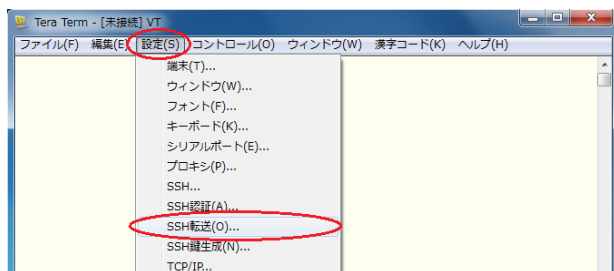
- (4) 下図のような画面でファイルを転送できるようになります。



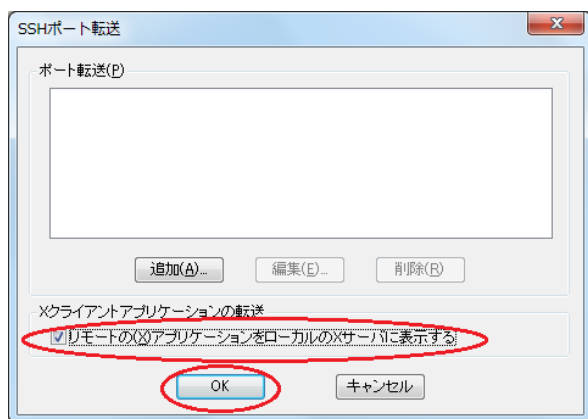
2.1.2.3. X Window System を利用するアプリケーションの使用

X Window System を利用するアプリケーションを使用する場合は事前に本手順を実行します。

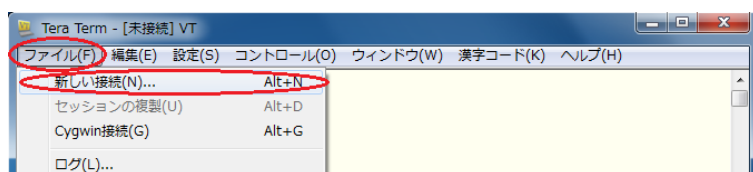
- (1) SSL-VPN 接続を開始します。接続方法は『SSL-VPN 利用者マニュアル』 (<http://www.j-focus.jp/sslvpn/>) を参照します。
- (2) お手元のマシン (localhost) に Cygwin/X、Xming 等の X サーバソフトウェアをあらかじめインストールしておき、お手元のマシン (localhost) で X サーバを起動します。
- (3) スタートメニュー [すべてのプログラム] → [Tera Term] を順に選択します。
- (4) Tera Term 『新しい接続の設定』画面で、[キャンセル] ボタンをクリックします。
- (5) Tera Term メニュー [設定] → [SSH 転送] を順に選択します。



- (6) 『SSH ポート転送』画面で、[リモートの (x) アプリケーションをローカルの X サーバに表示する] にチェックを入れ、[OK] ボタンをクリックします。



- (7) Tera Term メニュー [ファイル] → [新しい接続] を順に選択します。



- (8) 前述の手順『2.1.1.2. ログインサーバへの SSH 接続』、『2.1.1.3. 共用フロントエンドサーバへの接続』を使って、共用フロントエンドサーバに SSH 接続を行い、続けてアプリケーション固有の操作を行うことで、X Window System を利用するアプリケーションを使用できます。

2.1.3. 高度計算科学研究支援センター内でのログイン方法

前述『2.1.2. インターネットからの SSL-VPN 接続による利用方法』の各小節(2)以降の手順を使って、センター内から FOCUS スパコンシステムにログインできます。なお、講習用端末のハードディスクは、再起動すると初期化されますのでご注意ください。

2.2. パスワードの変更

共用フロントエンドサーバにログインするときに使うパスワードを変更する手順を記載します。

なお、インターネットからログインサーバに接続するときに使用する、公開鍵・秘密鍵のパスフレーズは、お手持のマシンで入力されたフレーズです（本手順で扱うパスワードとは異なります）。

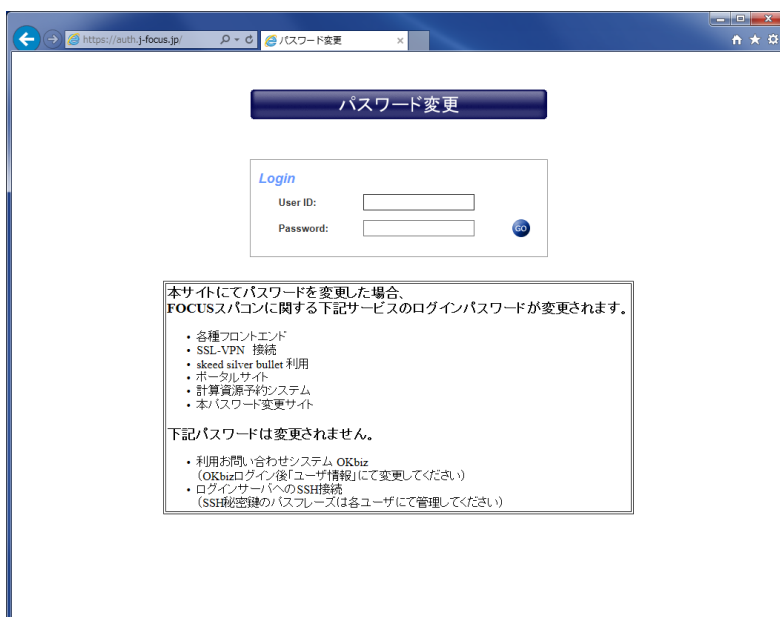
※他のパスワードを変更するときは、それぞれ以下の箇所を参照してください。

- ・アカウントのパスワード : 2.2.1.パスワードの変更（センター内）
- ・公開鍵・秘密鍵のパスフレーズ : 2.1.3.1.事前準備（秘密鍵・公開鍵の作成と登録）
- ・OKBiz のパスワード : 『利用上のご質問お問い合わせシステム利用方法（FOCUS OKBiz）』
(http://www.j-focus.jp/faq_guide/)

手順は次の通りです。

2.2.1. パスワードの変更（センター外）

- (1) SSL-VPN 接続を開始します。接続方法は『SSL-VPN 利用者マニュアル』(<http://www.j-focus.jp/sslvpn/>)を参照します。
- (2) Web ブラウザで以下の URL に接続して、パスワードの変更を行います。
[URL] <https://auth.j-focus.jp/>



2.2.2. パスワードの変更（センター内）

センター内のネットワークからパスワードを変更する場合は、上記『2.2.1. パスワードの変更（センター外）』の(2)を実施します。

2.3. ログインシェル

初期登録時、各システムのシェルはbash を使用可能としています。

ご参考までに他のシェルを利用される場合に用意すべきファイルを以下に記載します。bash 以外のシェルの環境設定ファイルは初期登録時には用意されていません。

表 2.3 環境設定ファイル

環境設定ファイル名	利用ログインシェル			
	bash	sh	ksh	csh / tcsh
.bash_profile	○			
.bashrc	○	○		
.profile		○	○	
.cshrc				○
.login				○

ログインシェル環境を変更されたい場合は、OKBiz (<https://secure.okbiz.okwave.jp/focus/>) でお問い合わせください。なお、お問い合わせの際はアカウント名（「u」 + 「課題名」 + 数字 4 桁）の記載をお願いします。

2.4. 追加ストレージ領域(/home3, Luster File System)の利用方法

- 利用には追加契約が必要となります。月単位・10GB 単位で容量を課題単位で追加できます。希望される場合は OKBiz にて契約容量変更希望日の 2 業務日前 9:00 までにご連絡ください。
- 追加ストレージ領域の契約を終了する場合、0GB への容量変更を申請していただくか、新規追加または容量変更申請時に利用終了日をご指定ください。
- 月単位契約の場合、特にお申し出がない限りは翌月以降も自動的に継続となりますことご注意ください。

【ハードリミット設定について】

システム上でのハードリミットは契約容量の 2 倍に設定します。

/home3 の使用量が契約容量に達したとしても、猶予期間（1 週間）の間は/home3 へ契約量の 2 倍まで書込みを行うことができます。

使用量が契約容量未満になれば猶予期間はリセットされます。

/home3 の使用量が契約容量を超えても、猶予期間内に契約容量を変更することで/home3 への書込みを止めることなくご利用いただけます。

猶予期間を過ぎてても/home3 の使用量が契約容量を超えていた場合は、使用量が契約容量未満となるまで/home3 への書込みができなくなります。

容量超過していたとしても、超過分のデータが自動的に削除されることはありません。

【契約超過分課金について】

契約容量を超えるデータを長期間(1 か月以上)置いていた場合は、超過分について 10GB 単位で課金いたします。

例) 変更後の契約容量が 300GB であっても、実利用量が 495GB の場合、

契約容量を 500GB とみなして課金します。

/home3 の使用量が契約容量を超えた際は、お早めに契約容量の変更やデータ退避等の対応をお願いいたします。

【利用終了時の注意点】

/home3 の契約終了（0GB への容量変更）の場合、契約終了月の月末をもってデータは削除されます。

2.4.1. Lustre File System 環境構成概要

本節では以下の略語を使用しています。

MDS : Meta Data Server の略

MDT : Meta Data Target の略

OSS : Object Storage Server の略

OST : Object Storage Target の略

Lustre File System の構成概要を示します。

ID	Filesystem Name	Mount Point	INET	Size	i-node 数	MDS	OSS	MDT/OST	OST 数	Stripe Count	Stripe Size
1	home3	/home3	@o2ib	458.54 TiB	2,199,486,464	home3srv1-ib (home3srv2-ib)	home3srv1-ib home3srv2-ib	home3stg	4	1	1,048,576Bytes

(※) MDS における () は standby node を意味します。

2.4.2. /home3 利用状況の確認

/home1 の容量は課題 (グループ) あたり 200GB ですが、追加ストレージ領域 (/home3) の容量についてはご契約内容により異なります。

現在のクォータ値 (契約容量) 等利用状況を確認するには `lfs` コマンドを使用します。

[コマンド]

```
lfs quota -g グループ名 /home3
```

[実行結果]

```
$ lfs quota -g gxxx /home3
```

```
Disk quotas for group gxxx (gid xxxx):
```

Filesystem	kbytes	quota	limit	grace	files	quota	limit	grace
/home3	59524	1048576	2097152	-	458	0	0	-

<項目説明 (左から) >

kbyte : 使用量(kbytes)

quota : 契約容量(ソフトリミット) (kbytes)

limit : 最大容量(ハードリミット) (kbytes)

grace : 契約容量越えの許容期間

files : 使用中のファイル数

2.4.3. Stripe Size/Stripe Count

ディレクトリおよびファイルの I/O Stripe 設定方法について記載します。

Lustre File System では各ファイルを格納する Stripe Count (OST の数) および Stripe Size をディレクトリやファイルごとに設定することができます。

デフォルトでは Stripe 設定は無しとなっていますので、各ファイルは1つの OST へ格納されます。

2.4.3.1. Stripe 設定方法 (lfs setstripe)

Stripe の設定は lfs setstripe コマンドで実施します。

基本的な使用方法是以下となります。

```
lfs setstripe [--size|-s stripe_size] [--offset|-o start_ost] [--count|-c stripe_count] [--pool|-p pool_name] <filename|dirname>
```

- stripe_size:
 - 各 OST を Stripe する際の Stripe Size を指定
 - k, m, g でそれぞれ Kbytes, Mbytes, Gbytes 指定が可能
 - Default は 0
- start_ost:
 - Stripe を開始する OST を指定
 - Default は -1 で Random
- stripe_count:
 - 使用する OST の数を指定
 - 1 で全 OST
 - Default は 0
- pool_name
 - 使用するプール名を指定

例 1) 2OST, Stripe Size=1MBytes で Stripe を設定

```
$ mkdir /home3/ddn/testdir/st2
$ lfs setstripe -s 1m -c 2 /home3/ddn/testdir/st2
```

例 2) 全 OST, Stripe Size=4MBytes で Stripe を設定

```
$ mkdir /home3/ddn/testdir/st_all
$ lfs setstripe -s 4m -c -1 /home3/ddn/testdir/st_all
```

2.4.3.2. Stripe 確認方法 (lfs getstripe/lfs osts)

Stripe の確認は lfs getstripe コマンドで実施します。
基本的な使用法は以下となります。

```
lfs getstripe [--obd|-O <uuid>] [--quiet|-q] [--verbose|-v] [--recursive|-r] <filename|dirname>
```

- **--obd|-O <uuid>:**
指定した OST に属するファイルを表示
- **--quiet|-q:**
出力項目の制限
- **--verbose|-v:**
Verbose Mode
- **--recursive|-r:**
Recursive Mode

例1) 下記において testfile1 は obdidx で示される 2 つの OST に Stripe されていることを確認することができます。

```
$ lfs getstripe /home3/ddn/testdir/st2/testfile1
/home3/ddn/testdir/st2/testfile1
lmm_stripe_count: 2
lmm_stripe_size: 1048576
lmm_stripe_offset: 54
      obdidx      objid      objid      group
          7        258      0x102         0
          14        258      0x102         0

$
```

各 obdidx の Object Name は lfs osts の出力結果から確認することができます。

```
$ lfs osts
OBDS::
0: home3-OST0000_UUID ACTIVE
1: home3-OST0001_UUID ACTIVE
2: home3-OST0002_UUID ACTIVE
3: home3-OST0003_UUID ACTIVE

$
```

例2) `-r` を付与すると recursive mode となり指定したディレクトリ配下に存在する全てのオブジェクトについて Stripe 情報が表示されます。

```
$ lfs getstripe -r /home3/ddn/testdir/
/home3/ddn/testdir/
stripe_count:  1 stripe_size:  1048576 stripe_offset: -1
/home3/ddn/testdir/st2
stripe_count:  2 stripe_size:  1048576 stripe_offset: -1
/home3/ddn/testdir/st2/testfile1
lmm_stripe_count:  2
lmm_stripe_size:  1048576
lmm_stripe_offset: 54
      obdidx      objid      objid      group
        7         258      0x102         0
       14         258      0x102         0

/home3/ddn/testdir/st2/testfile2
lmm_stripe_count:  2
lmm_stripe_size:  1048576
lmm_stripe_offset: 66
      obdidx      objid      objid      group
        1         258      0x102         0
        8         258      0x102         0

/home3/ddn/testdir/st2/testfile3
lmm_stripe_count:  2
lmm_stripe_size:  1048576
lmm_stripe_offset: 13
      obdidx      objid      objid      group
       11         258      0x102         0
       16         258      0x102         0

/home3/ddn/testdir/st_all
stripe_count:  -1 stripe_size:  4194304 stripe_offset: -1
/home3/ddn/testdir/st_all/testfile1
lmm_stripe_count:  72
lmm_stripe_size:  4194304
lmm_stripe_offset: 31
      obdidx      objid      objid      group
        2         258      0x102         0
        4         258      0x102         0
        8         258      0x102         0
       11         258      0x102         0
       15         258      0x102         0
       17         258      0x102         0
<省略>
$
```

例 3) 特定の OST に属するファイルを探す場合は下記を実施します。

```
$ lfs getstripe --r --obd home3-OST0000_UUID /home3/ddn/testdir/

/home3/ddn/testdir/st_all/testfile1
lmm_stripe_count: 72
lmm_stripe_size: 4194304
lmm_stripe_offset: 31
      obdidx      objid      objid      group
          0        258      0x102        0 *

/home3/ddn/testdir/st_all/testfile2
lmm_stripe_count: 72
lmm_stripe_size: 4194304
lmm_stripe_offset: 37
      obdidx      objid      objid      group
          0        259      0x103        0 *

/home3/ddn/testdir/st_all/testfile3
lmm_stripe_count: 72
lmm_stripe_size: 4194304
lmm_stripe_offset: 43
      obdidx      objid      objid      group
          0        260      0x104        0 *

$
```

2.5. ~~クラウドストレージの利用方法~~（提供終了）

FOCUS スパコンシステムにおけるクラウドストレージシステムの提供は、2018 年 3 月に終了しました。

2.6. 改行コード

UNIX/Linux 系 OS (RHEL6/CentOS6) と Windows 系 OS では、改行コードに違いがあります。

2.6.1. 改行コード

改行コードには、LF (Line Feed)、CR (Carriage Return) があり、UNIX/Linux 系では LF、Windows 系では CR+LF となります。

表 2.6.1 OS 改行コード

OS	改行コード
RHEL6、CentOS6	LF
Windows 系 (7/8 など)	CR+LF

2.6.2. エディタ

(1) emacs (RHEL6/CentOS6)

emacs では、CR+LF、LF 両方のテキストファイルを編集することができます。emacs で CR+LF を編集した場合、次の通り " (DOS) " と表示されます。

```
----- (DOS) ----F1  dos.c                (C Abbrev) --L1--All-----
```

(2) vi / vim (RHEL6/CentOS6)

RHEL6/CentOS6 の vi (vim) では、CR+LF、LF 両方のテキストファイルを編集することができます。vi (vim) で CR+LF を編集した場合、 "[dos]" と表示されます。

2.6.3. 改行コードの変換

改行コードの変換方法を示します。

(1) LF から CR+LF への変換

```
$ nkf -Lw UNIX/Linux 系ファイル > output ファイル
```

(2) CR+LF から LF への変換

```
$ nkf -Lu Windows 系ファイル > output ファイル
```


2.7. module コマンド

FOCUS スパコンでは、様々なプログラムの実行環境の設定に module コマンドを用います。

(1) 対応している環境の一覧表示

```
$ module avail
```

【実行例】

```
$ module avail
----- /home1/share/modulefiles -----
MIZUHO_ABINIT-MP3.01+impi-4.1.1
MIZUHO_ABINIT-MP3.0_FOCUS+impi-4.1.1
MPI-mpi-17.1.132
MPI-mpi-17.6.256
MPI-mpi-18.3.222
MPI-nmpi-1.1.1
MPI-nmpi-1.3.0
MPI-openmpi-1.10.7+Intel-17.0.1.132
MPI-openmpi-1.10.7+gnu-4.4.7+cuda-8.0
MPI-openmpi-1.10.7+gnu-6.3.0
MPI-openmpi-2.1.1+gnu-6.3.0
MPI-openmpi-2.1.1+gnu-6.3.0+cuda-8.0
MPI-openmpi-2.1.1+intel-17.0.1.132+cuda-8.0
MPI-openmpi-2.1.3+gnu-4.4.7
MPI-openmpi-2.1.3+gnu-4.8.2
MPI-openmpi-2.1.3+intel-17.0.1.132
Math-MKL-17.1.132+intel-17.0.1.132
Math-MKL-17.6.256+intel-17.0.6.256
Math-MKL-18.3.222+intel-18.0.3.222
<...snip...>
```

(2) 環境設定の読み込み

```
$ module load モジュール名
```

【実行例】

```
$ module load PrgEnv-intel-18.0.3.222
```

(3) 読み込んだ環境設定の表示

```
$ module list
```

【実行例】

```
$ module list
Currently Loaded Modulefiles:
  1) PrgEnv-intel-18.0.3.222
```

(4) 環境設定の解除

```
$ module unload モジュール名
```

【実行例】

```
$ module unload PrgEnv-intel-18.0.3.222
$ module list
No Modulefiles Currently Loaded.
```

読み込むモジュールのバージョンについての注意事項

開発時に読み込んだモジュールと実行時に読み込んだモジュールのバージョンが異なる場合、実行に失敗したり意図した動作にならない場合がありますので、適切なバージョンのモジュールを読み込むようご注意ください。

3. コンパイラ、MPI の使用方法

3.1. Intel コンパイラ

コンパイル環境として「インテル® Parallel Studio XE」を利用することができます。
 なお、現在の設定環境は次のように module list コマンドで確認します。

【設定環境の確認例－初期状態】

```
$ module list
No Modulefiles Currently Loaded.
$
```

【設定環境の確認例－設定状態】

```
$ module list
Currently Loaded Modulefiles:
  1) PrgEnv-intel-18.0.3.222 2) MPI-impi-18.3.222
```

←Intel コンパイラと Intel MPI 環境が
設定されている状態の例です。

\$

3.1.1. コンパイラ環境の設定 (Intel コンパイラ)

1. 環境の設定

```
$ module load PrgEnv-intel-18.0.3.222
```

※ MKL (Math Kernel Library) のリンク環境も同時に設定されます。

2. 設定環境の確認

```
$ module list
Currently Loaded Modulefiles:
  1) PrgEnv-intel-18.0.3.222
```

3. 環境の設定解除

```
$ module unload PrgEnv-intel-18.0.3.222
```

3.1.2. コンパイルコマンド (Intel コンパイラ)

各言語のコンパイルコマンドは以下の通りです。

使用言語	コマンド	コマンド形式
Fortran	ifort	ifort [オプション] ファイル...
C 言語	icc	icc [オプション] ファイル...
C++	icpc	icpc [オプション] ファイル...

3.1.3. MPI 環境の設定 (Intel コンパイラ)

3.1.3.1. Open MPI

1. 環境の設定

```
$ module load MPI-openmpi-2.1.3+intel-17.0.1.132
```

2. 設定環境の確認

```
$ module list
Currently Loaded Modulefiles:
  1) MPI-openmpi-2.1.3+intel-17.0.1.132
```

3. 環境の設定解除

```
$ module unload MPI-openmpi-2.1.3+intel-17.0.1.132
```

4. コンパイルコマンド

各言語のコンパイルコマンドは以下の通りです。

使用言語	コマンド	コマンド形式
MPI (Fortran)	mpif90	mpif90 [オプション] ファイル...
MPI (C 言語)	mpicc	mpicc [オプション] ファイル...
MPI (C++)	mpicxx	mpicxx [オプション] ファイル...

3.1.3.2. MPICH2

現在 MPICH2 環境の module は提供していません。

3.1.3.3. Intel MPI

1. 環境の設定

```
$ module load MPI-impi-18.3.222
```

2. 設定環境の確認

```
$ module list
Currently Loaded Modulefiles:
  1) MPI-impi-18.3.222
```

3. 環境の設定解除

```
$ module unload MPI-impi-18.3.222
```

4. コンパイルコマンド

各言語のコンパイルコマンドは以下の通りです。

使用言語	コマンド	コマンド形式
MPI (Fortran)	mpiifort	mpiifort [オプション] ファイル...
MPI (C 言語)	mpiicc	mpiicc [オプション] ファイル...
MPI (C++)	piicpc	mpiicpc [オプション] ファイル...

FOCUS スパコン環境では旧バージョンの Intel MPI も提供しておりますが、旧バージョンには不具合がある場合があります。

特にバージョンを指定する必要が無い場合は、FOCUS スパコン環境で提供されている最新版の Intel MPI をご利用下さい。

3.1.4. コンパイラ、MPI 環境の切替え

コンパイラ、MPI 環境を切り替えるコマンドは以下の通りです。

コンパイラ	MPI	環境設定コマンド	設定解除コマンド
Intel コンパイラ	OpenMPI	module load PrgEnv-intel-17.0.1.132 module load MPI-Openmpi-2.1.3+intel-17.0.1.132	module unload PrgEnv-intel-17.0.1.132 module unload MPI-openmpi-2.1.3+intel-17.0.1.132
	Intel MPI	module load PrgEnv-intel-18.0.3.222 module load MPI-impi-18.3.222	module unload PrgEnv-intel-18.0.3.222 module unload MPI-impi-18.3.222

3.1.5. コンパイル・オプション (Intel コンパイラ)

Intel コンパイラの主なコンパイル・オプションを示します。

オプション	説明
-c	オブジェクトファイルが生成された後、コンパイル処理を停止します。コンパイラは、C または C++ の各ソースファイルまたは前処理されたソースファイルからオブジェクトファイルを生成します。
-C	前処理されたソースの出力にコメントを配置します。
-o filename	実行可能ファイル名を指定します。省略時は a.out で作成されます。
-g	一般的な開発環境のデバッガーで使用できるデバッグ情報を生成します。このオプションは、/O2 (-O2) (または別の O オプション) が指定されない限り、/O2 (-O2) をオフにして /Od (-O0) をデフォルトにします。
-O0	最適化は行われません。このオプションは、アプリケーション開発の初期段階およびデバッグ時に使用します。アプリケーションが正常に動作することを確認した後は、より高度なオプションを使用してください。
-O1	サイズを考慮した最適化を行います。オブジェクトのサイズを増やす傾向がある最適化を省略します。多くの場合、最小限のサイズで最適化されたコードが作成されます。コードサイズが大きいため、メモリーページングが問題になる巨大なサーバ/データベース・アプリケーションにおいて、このオプションは効果的です。
-O2	最速化します (デフォルト設定)。ベクトル化と実行速度を改善する多くの最適化を有効にします。多くの場合、/O1 (-O1) よりも速いコードを作成します。
-O3	/O2 (-O2) の最適化に加えて、スカラ置換、ループアンロール、分岐を除去するコード反復、効率的にキャッシュを使用するループ・ブロッキング、データ・プリフェッチ機能など、強力なループ最適化およびメモリーアクセス最適化を行います。 /O3 (-O3) オプションは、特に浮動小数点演算を多用するループや大きなデータセットを処理するループを含むアプリケーションに推奨します。この最適化は、場合によって /O2 (-O2) の最適化よりもアプリケーションの実行が遅くなることがあります。
-fast	プログラム全体の速度を最大限にします。次のオプションを設定します。 -ipo、-O3、-no-prec-div、-static、-xHost
-static	静的ライブラリをリンクします。
-opt-report [n]	最適化レポートを作成し、stderr 出力します。n には、0 (レポートなし) から 3 (最大限の情報) の範囲で詳細レベルを指定します。デフォルトは 2 です。
-openmp	OpenMP* 指示句がある場合、その指示によるマルチスレッド・コードが生成されます。スタックのサイズを増やさなければならないことがあります。
-parallel	自動パラライザは、安全に並列実行可能な構造のループ (インテル® Cilk™ Plus のアレイ・ノテーションによって暗黙的に定義されるループを含む) を検出し、そのループに対するマルチスレッド・コードを自動生成します。
-par-report[n]	自動並列化の診断レベルを制御します。n には、0 (レポートなし) から 3 (最大限の情報) の範囲で詳細レベルを指定します。デフォルトは 0 です。
-ip	単一ファイルの最適化を行います。現在のソースファイルを対象にしたインライン展開を含むプロシージャーク間の最適化です。
-ipo[n]	インライン展開およびその他のプロシージャーク間の最適化を複数のソースファイルに対して行います。オプションの n 引数には、コンパイル時に生成するオブジェクトファイルの最大数を指定します。デフォルトの n は 0 です (コンパイラが最適なファイル数を自動選択)。 警告: 条件によってはコンパイル時間とコードサイズが大幅に増加する場合があります。
-prof-gen	プロファイル最適化で参照する動的なパフォーマンス・データを生成するため、プログラ

	ムにインストルメント・コードを埋め込みます。
-prof-use	最適化中に prof-gen オプションで生成した実行ファイルのプロファイリング情報を参照します。
-prof-dir dir	プロファイル出力ファイル*.dyn および*.dpi を格納するディレクトリを指定します。
-fp-model name	浮動小数点演算における演算モデルを制御します。特定の最適化を制限して浮動小数点結果の一貫性を強化します。name の値は次のとおりです。 fast=[1 2] - 精度や一貫性を多少低くすることにより、さらに強力な最適化が可能になります (デフォルトは、fast=1)。一部の最適化は、インテル® マイクロプロセッサのみに適用される場合があります。 precise - 精度に影響しない最適化のみ有効にします。 double/extended/source - 中間結果をそれぞれ倍精度、拡張精度、ソースの精度で丸めます。変更されない限り、precise も適用されます。インテル® Fortran コンパイラでは、double オプションおよび extended オプションは利用できません。 except - 浮動小数点例外セマンティクスを使用します。 strict - precise オプションと except オプションの両方を有効にし、デフォルトの浮動小数点環境を想定しません。 推奨：浮動小数点演算の一貫性や再現性が重要な状況では、/fp:precise /fp:source (-fp-model precise -fp-model source) を推奨します。
-[no]restrict	restrict キーワードとともに指定すると、ポインターの一義化が有効[無効]になります。デフォルトではオフです (C/C++)。
-mkl [=parallel, sequential, cluster]	数値演算ライブラリ Intel Math Kernel Library (MKL) をリンクします。最適化された BLAS, LAPACK, ベクトル演算ライブラリなどを利用する場合に用います。 -mkl -mkl=parallel 並列のインテル® MKL ライブラリを使用 -mkl=sequential シーケンシャルなインテル® MKL ライブラリを使用 -mkl=cluster クラスタのインテル® MKL ライブラリを使用
-mcmmodel [=small, medium, large]	コードとデータサイズを指定します。 -mcmmodel=small : デフォルト、コードとデータサイズが 2GB までの制限あり -mcmmodel=medium : コードが 2GB までの制限あり、データサイズは制限なし -mcmmodel=large : コードとデータサイズに制限なし

3.1.6. コンパイル方法 (Intel コンパイラ)

主なコンパイル方法を以下に示します。

3.1.6.1. 逐次プログラム

1. Fortran の例

```
$ ifort test.f90
```

2. C 言語の例

```
$ icc test.c
```

3. C++の例

```
$ icpc test.cpp
```

3.1.6.2. 自動並列プログラム

1. Fortran の例

```
$ ifort -parallel test.f90
```

2. C 言語の例

```
$ icc -parallel test.c
```

3. C++の例

```
$ icpc -parallel test.cpp
```

3.1.6.3. OpenMP プログラム

1. Fortran の例

```
$ ifort -openmp test.f90
```

2. C 言語の例

```
$ icc -openmp test.c
```

3. C++の例

```
$ icpc -openmp test.cpp
```

3.1.6.4. MPI プログラム (Intel MPI)

1. Fortran の例

```
$ mpiifort test.f90
```

2. C 言語の例

```
$ mpiicc test.c
```

3. C++の例

```
$ mpiicpc test.cpp
```

3.1.6.5. MPI プログラム (OpenMPI)

1. Fortran の例

```
$ mpif90 test.f90
```

2. C 言語の例

```
$ mpicc test.c
```

3. C++の例

```
$ mpicxx test.cpp
```


3.1.6.6. MKL (Math Kernel Library)

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・ライブラリ関数、乱数生成関数を利用することができます。

前述の手順『3.1.1 コンパイラ環境の設定 (Intel コンパイラ)』で、MKL のリンク環境も同時に設定されます。

以下に C 言語のコンパイルの例を示します。C 言語 (icc, mpicc) のみを記述していますが、Fortran (ifort, mpiifort)、C++ (icpc, mpiicpc) はコマンドを読み替えてください。

1. 並列/動的ライブラリ

```
$ icc test.c -mkl
または
$ icc test.c -mkl= parallel
```

2. 並列/静的ライブラリ

```
$ icc test.c -mkl -static-intel
または
$ icc test.c -mkl= parallel -static-intel
```

3. シーケンシャル/動的ライブラリ

```
$ icc test.c -mkl=sequential
```

4. シーケンシャル/静的ライブラリ

```
$ icc test.c -mkl=sequential -static-intel
```

5. MPI/動的ライブラリ

```
$ mpicc mpisample.c -mkl=cluster
```

6. MPI/静的ライブラリ

```
$ mpicc mpisample.c -mkl=cluster -static-intel
```

3.1.7. コンパイル時の注意点 (Intel コンパイラ)

各システム毎でプロセッサのアーキテクチャが異なります。コンパイル時に以下のオプション指定に注意してください。

オプション	A システム	B システム	C システム	D システム	E システム	F システム	G システム	H システム	V システム
-fast	×	×	×	○	○	○	○	○	○
-xHOST	×	×	×	○	○	○	○	○	○
-xSSE4.2	○	○	○	○	○	○	○	○	○
-xAVX	×	×	×	○	○	○	○	○	○
-xCORE-AVX-I	×	×	×	○	○	○	×	○	○
-xCORE-AVX2	×	×	×	×	×	○	×	○	○
上記オプション無し	○	○	○	○	○	○	○	○	○

×：指定すると動作しない

○：動作する

3.2. GNU コンパイラ

コンパイル環境として GCC 4.4.7 と GCC 6.3.0 を利用することができます。

標準で GCC 4.4.7 が利用できます。GCC 6.3.0 を利用するときは以下の 3.2.1 の手順を実施して、環境を設定します。

3.2.1. コンパイラ環境変数の設定 (GNU コンパイラ)

1. 環境の設定

```
$ module load PrgEnv-gnu-6.3.0
```

2. 設定環境の確認

```
$ module list
Currently Loaded Modulefiles:
  1) PrgEnv-gnu-6.3.0
```

3. 環境の設定解除

```
$ module unload PrgEnv-gnu-6.3.0
```

3.2.2. コンパイルコマンド (GNU コンパイラ)

各言語のコンパイルコマンドは以下の通りです。

使用言語	コマンド	コマンド形式
Fortran	gfortran	gfortran [オプション] ファイル...
C 言語	gcc	gcc [オプション] ファイル...
C++	g++	g++ [オプション] ファイル...

3.2.3. MPI 環境変数の設定 (GNU コンパイラ)

3.2.3.1. Open MPI

1. 環境の設定

```
$ module load MPI-openmpi-2.1.1+gnu-6.3.0
```

2. 設定環境の確認

```
$ module list  
Currently Loaded Modulefiles:  
  1) MPI-openmpi-2.1.1+gnu-6.3.0
```

3. 環境の設定解除

```
$ module unload MPI-openmpi-2.1.1+gnu-6.3.0
```

4. 各言語のコンパイルコマンドは以下の通りです。

使用言語	コマンド	コマンド形式
MPI (Fortran)	mpif90	mpif90 [オプション] ファイル...
MPI (C 言語)	mpicc	mpicc [オプション] ファイル...
MPI (C++)	mpicxx	mpicxx [オプション] ファイル...

3.2.3.2. MPICH2

現在MPICH2 環境の module は提供していません。

3.2.4. コンパイラ、MPI 環境の切替え

コンパイラ、MPI 環境を切り替えるコマンドは以下の通りです。

コンパイラ	MPI	環境設定コマンド	設定解除コマンド
標準 GNU (GNU 4.4.7)	OpenMPI	module load MPI-openmpi-2.1.3+gnu-4.4.7	module unload MPI-openmpi-2.1.3+gnu-4.4.7
GNU6.3.0	OpenMPI	module load PrgEnv-gnu-6.3.0 module load MPI-openmpi-2.1.1+gnu-6.3.0	module unload PrgEnv-gnu-6.3.0 module unload MPI-openmpi-2.1.1+gnu-6.3.0

3.2.5. コンパイル・オプション (GNU コンパイラ)

GNU コンパイラの主なコンパイル・オプションを示します。

オプション	説明
-c	ソースファイルのコンパイル、または、アセンブルを行います。リンクは行いません。
-o file	出力先を引数 file に指定します。このオプションは GCC が実行可能ファイル、オブジェクトファイル、アセンブラファイル、プリプロセス済み C コードなどの、いかなる種類の出力を行なう場合にも適用可能です。出力ファイルは 1 つしか指定できないため、'-o' を複数の入力ファイルをコンパイルする際に使用することは、実行ファイルを出力する時以外は無意味です。'-o' オプションを指定しなかった場合のデフォルトは、実行ファイルを作る場合は 'a.out' という名前であり、'source.suffix' の形式のファイル名を持ったソースファイルのオブジェクトファイルは 'source.o' であり、アセンブラのファイルは 'source.s' です。プリプロセス済みの C 言語は、全て標準出力に送られます。
-I	インクルードファイルのパスを指定します。
-g	デバッグ用の情報を保存します。
-l library	名前が library であるライブラリをリンク時に使用します。
-static	ダイナミックリンクをサポートするシステムにおいて、このオプションは共有ライブラリとのリンクを抑制します。
-O2	高度な最適化を行います。サポートされている最適化手段のうち、空間と速度のトレードオフを含まないものはほとんど使用されます。例えばループのアンローリングや関数のインライン化は行われません。-O と比較して、このオプションはコンパイル時間と生成コードの性能の双方を増加させます。
-O3	さらなる最適化を行います。これは -O2 が行う全ての最適化手段に加えて -finline-functions も有効にします
-O0	最適化を行いません。

3.2.6. コンパイル方法 (GNU コンパイラ)

主なコンパイル方法を以下に示します。

3.2.6.1. 逐次プログラム

1. Fortran の例

```
$ gfortran test.f90
```

2. C 言語の例

```
$ gcc test.c
```

3. C++ の例

```
$ g++ test.cpp
```

3.2.6.2. OpenMP プログラム

1. Fortran の例

```
$ gfortran -fopenmp test.f90
```

2. C 言語の例

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

3. C++ の例

```
$ g++ -fopenmp test.cpp
```

3.2.6.3. MPI プログラム (OpenMPI、MPICH2、Intel MPI)

1. Fortran の例

```
$ mpif90 test.f90
```

2. C 言語の例

```
$ mpicc test.c
```

3. C++ の例

```
$ mpiCC test.cpp
```

3.2.7. コンパイル時の注意点 (GNU コンパイラ)

各システム毎でプロセッサのアーキテクチャが異なります。コンパイル時に以下のオプション指定に注意してください。

GCC 4 の場合

オプション	A システム	B システム	C システム	D システム	E システム	F システム	G システム	H システム	V システム
-march=corei7	○	○	○	○	○	○	○	○	○
-march=corei7-avx	×	×	×	○	○	○	○	○	○
-march=corei-avx-i	×	×	×	○	○	○	×	○	○
-march=corei-avx2	×	×	×	×	×	○	×	○	○
上記オプション無し	○	○	○	○	○	○	○	○	○

×：指定すると動作しない

○：動作する

GCC 6 の場合

オプション	A システム	B システム	C システム	D システム	E システム	F システム	G システム	H システム	V システム
-march=westmere	○	○	○	○	○	○	○	○	○
-march=nehalem	×	○	×	○	○	○	○	○	○
-march=sandybridge	×	×	×	○	○	○	○	○	○
-march=ivybridge	×	×	×	○	○	○	×	○	○
-march=broadwell	×	×	×	×	×	○	×	○	○
上記オプション無し	○	○	○	○	○	○	○	○	○

×：指定すると動作しない

○：動作する

4. プログラムの実行方法

ジョブスケジューリングには「SLURM (Slurm Workload Manager)」を採用しています。

4.1. キュー

4.1.1. キューの一覧

フロントエンドサーバから演算ノードに対して、ジョブを投入できるキューの一覧を示します。

表 4.1.1 キューの一覧

キュー名	実行システム	最長実行時間	最大同時ジョブ投入数	最大ノード数 (※1)	ジョブあたり 最大ノード数 (※1)	備考
a024h	A	24 時間	2000/利用者	208	208	
b168h	A	168 時間	2000/利用者	32	32	
b024h	B	24 時間	2000/利用者	2	2	
b168h	B	168 時間	2000/利用者	1	1	
d024h	D	24 時間	2000/利用者	80	80	下記 (4) 参照
d168h	D	168 時間	2000/利用者	40	40	下記 (4) 参照
e168h	E	168 時間	2000/利用者	24	24	※3, Phi 1 基, 下記 (5) 参照
f024h	F	24 時間	2000/利用者	60	60	
f072h	F	72 時間	2000/利用者	32	32	
f072h_p100	F	72 時間	2000/利用者	2	2	GPU 搭載
h024h	H	24 時間	2000/利用者	136	136	
h072h	H	72 時間	2000/利用者	102	102	
v024h	V	24 時間	2000/利用者	2	2	
v072h	V	72 時間	2000/利用者	2	2	
p072h	HPCI プリポスト	72 時間	3/利用者	1	1	無償
a006m	A	6 分	5/利用者	2	2	※2
b006m	B	6 分	5/利用者	2	2	※2
d006m	D	6 分	5/利用者	2	2	※2, 下記 (4) 参照
e006m	E	6 分	5/利用者	2	2	※2, ※3, Phi 1 基, 下記 (5) 参照
f006m	F	6 分	5/利用者	2	2	※2, GPU 搭載
g006m	G	6 分	5/利用者	4	4	※2, 下記 (3) 参照
h006m	H	6 分	5/利用者	4	4	※2
v006m	V	6 分	5/利用者	2	2	※2

※1 予約状況により、変動する場合があります。

※2 デバッグキューです。計算資源利用料金は発生しません。デバッグ(ソフトウェアの動作検証等)にご利用ください。
 なお、FOCUS スパコン従量利用ソフトウェア (Gaussian09 等) を実行した場合、計算資源利用料金は発生しませんが、ソフトウェア利用料金は発生します。

※3 全ノード停止中のため利用を希望される場合は OKBiz (<https://secure.okbiz.okwave.jp/focus/>) にて技術質問からご相談ください。

- (1) 実際に本システムを利用する際には、`sinfo -s` コマンドで利用できるキュー名を確認して下さい。
- (2) キューのノード実行状況を確認する際には、`squeues` コマンドで確認して下さい。
- (3) デフォルトキューは「g006m」となります。
ジョブ実行時にキュー名が省略された場合「g006m」で実行されます。
- (4) 【H28.04.12 現在】下記のソフトウェアにて D システム上で、ジョブを実行した際にジョブが正常に実行されない場合があるとの報告があります。
各ソフトウェアをご利用の際は下記のとおりご対応ください。
 - ・ANSYS CFX : 回避策をベンダーにお問い合わせください。

- (5) 【H30.2.28 現在】下記のソフトウェアにて E システム上でジョブを実行した際にジョブが正常に実行されない場合があるとの報告があります。
以下のソフトウェアをご利用の際は下記のとおりご対応ください。

- ・ANSYS FLUENT : 回避策をベンダーにお問い合わせください。
- ・ANSYS CFX : E システムの利用を回避してください。
- ・STAR-CCM+ : `-mpi intel` オプションを指定してください。

その他ソフトウェアの E システムでの稼働状況につきましては提供ベンダーにお問い合わせください。

4.1.2. キュー情報の確認方法

ジョブ投入先のキュー名を確認するには、sinfo コマンドを実行します。

```
$ sinfo -s
```

【実行例】

```
$ sinfo -s
PARTITION AVAIL  TIMELIMIT  NODES (A/I/O/T)  NODELIST
a024h      up 1-00:00:00    0/194/11/205  a[001-095,097-204,207-208]
a168h      up 7-00:00:00      0/31/1/32  a[001-032]
b024h      up 1-00:00:00      0/2/0/2  b[001-002]
b168h      up 7-00:00:00      0/1/0/1  b001
d024h      up 1-00:00:00     17/63/0/80  d[001-080]
d168h      up 7-00:00:00     17/23/0/40  d[001-040]
e168h     inact 7-00:00:00      0/0/0/0
f024h      up 1-00:00:00     13/40/5/58  f[201-254,257-260]
f072h      up 3-00:00:00     13/18/1/32  f[201-232]
f072h_p100 up 3-00:00:00      0/2/0/2  f[601-602]
h024h      up 1-00:00:00    45/91/0/136  h[001-136]
h072h      up 3-00:00:00    45/57/0/102  h[001-102]
v024h      up 1-00:00:00      0/2/0/2  v[001-002]
v072h      up 3-00:00:00      0/2/0/2  v[001-002]
p072h      up 3-00:00:00      0/1/0/1  hpcipps1
g006m*     up      6:00      0/4/0/4  g[001-004]
a006m      up      6:00      0/2/0/2  a[207-208]
b006m      up      6:00      0/2/0/2  b[001-002]
d006m      up      6:00      0/2/0/2  d[079-080]
f006m      up      6:00      0/2/0/2  f[601-602]
h006m      up      6:00      0/4/0/4  h[068,102,131,136]
v006m      up      6:00      0/2/0/2  v[001-002]
```

<出力説明>

PARTITION	キュー名 (パーティション名)
AVAIL	キューの状態 (up or inact)
TIMELIMIT	最大実行時間
NODES (A/I/O/T)	ノードの状態 (allocated/idle/other/total)
NODELIST	キュー (パーティション) に割り当てられたノード

キューのノード実行状況を確認するには、squeues コマンドを実行します。
 注意：SLURM の標準コマンドの squeue とは別コマンドです。

```
$ sqqueues
```

【実行例】

```
$ sqqueues
QUEUE_NAME      TIMELIMIT      STATUS MAXNODES NNODES DEPEND PEND RUN FREE
a006m            6:00          up      2       0      0      0      0      2
a024h           1-00:00:00    up     194      0      0      0      0    194
a168h           7-00:00:00    up      31      0      0      0      0     31
b006m            6:00          up      2       0      0      0      0      2
b024h           1-00:00:00    up      2       0      0      0      0      2
b168h           7-00:00:00    up       1      0      0      0      0      1
d006m            6:00          up      2       0      0      0      0      2
d024h           1-00:00:00    up     80     17      0      0     17     63
d168h           7-00:00:00    up     40     17      0      0     17     23
e168h           7-00:00:00 inactive      0      0      0      0      0      0
f006m            6:00          up      2       0      0      0      0      2
f024h           1-00:00:00    up     53     13      0      0     13     40
f072h           3-00:00:00    up     31     13      0      0     13     18
f072h_p100       3-00:00:00    up      2       0      0      0      0      2
g006m            6:00          up      4       0      0      0      0      4
h006m            6:00          up      4       0      0      0      0      4
h024h           1-00:00:00    up    136     45      0      0     45     91
h072h           3-00:00:00    up    102     45      0      0     45     57
p072h           3-00:00:00    up       1      0      0      0      0      1
v006m            6:00          up      2       0      0      0      0      2
v024h           1-00:00:00    up      2       0      0      0      0      2
v072h           3-00:00:00    up      2       0      0      0      0      2
```

<出力説明>

QUEUE_NAME	キュー名 (パーティション名)
TIMELIMIT	最大実行時間
STATUS	キューの状態 (up or inact)
MAXNODES	最大ノード数
NNODES	実行中及び実行待ちのジョブが要求しているノード数
DEPEND	実行待ち (Dependency) のジョブが要求しているノード数
PEND	実行待ちのジョブが要求しているノード数
RUN	実行中のノード数
FREE	空きノード数

4.1.3. 利用可能なノード数の確認方法

空きノード数を確認する `freenodes` というコマンドを用意しています。ジョブ投入のための空きノード数の確認の目安にご利用ください。

```
$ freenodes
```

【出力形式】

```
Number of free nodes in A sys. with FDR-IB connected is 空きノード数 / 提供最大ノード数.
Number of free nodes in B sys. with FDR-IB connected is 空きノード数 / 提供最大ノード数.
Number of free nodes in D sys. with FDR-IB connected is 空きノード数 / 提供最大ノード数.
Number of free nodes in E sys. with FDR-IB connected is 空きノード数 / 提供最大ノード数.
Number of free nodes in F sys. with FDR-IB connected is 空きノード数 / 提供最大ノード数.
Number of free nodes in G sys. with 10GbE connected is 空きノード数 / 提供最大ノード数.
Number of free nodes in H sys. with 10GbE connected is 空きノード数 / 提供最大ノード数.
Number of free nodes in V sys. with FDR-IB connected is 空きノード数 / 提供最大ノード数.
Number of free nodes in HPCIPPS with FDR-IB connected is 空きノード数 / 提供最大ノード数.
```

【実行例】

```
$ freenodes
Number of free nodes in A sys. with QDR-IB connected is 195/ 195.
Number of free nodes in B sys. with QDR-IB connected is 2/ 2.
Number of free nodes in D sys. with FDR-IB connected is 63/ 80.
Number of free nodes in E sys. with FDR-IB connected is 0/ 0.
Number of free nodes in F sys. with FDR-IB connected is 42/ 55.
Number of free nodes in G sys. with 10GbE connected is 4/ 4.
Number of free nodes in H sys. with 10GbE connected is 92/ 136.
Number of free nodes in V sys. with FDR-IB connected is 2/ 2.
Number of free nodes in HPCIPPS with FDR-IB connected is 1/ 1.
```

4.2. ジョブの実行方法 (SLURM コマンド編)

ジョブを実行するには、フロントエンドサーバからコマンドを実行します。
ジョブ管理のためのコマンドは次のとおりです。

表 4.2 ジョブ管理コマンド一覧 (SLURM コマンド編)

コマンド	コマンド用途	手順
<code>sinfo -s</code>	キュー (パーティション) の情報を表示する	4.1.2. キュー情報の確認方法
<code>squeues</code>	キューのノード実行状況を表示する	
<code>freenodes</code>	空ノード数を表示する	4.1.3. 利用可能なノード数の確認方法
<code>sbatch</code>	ジョブを投入する	4.2.1. ジョブ投入コマンド (sbatch)
<code>squeue</code>	ジョブの状態を表示する	4.2.2. ジョブ情報表示コマンド (squeue)
<code>scancel</code>	ジョブをキャンセルする	4.2.3. ジョブのキャンセルコマンド (scancel)

4.2.1. ジョブ投入コマンド (sbatch)

ジョブを実行するために、ジョブ投入スクリプトを事前に作成します。sbatch コマンドにジョブ投入スクリプトを指定することで、ジョブがキューイングされ実行されます。

【sbatch コマンドの書式】

```
$ sbatch ジョブ投入スクリプト
```

注) bsub と異なり、リダイレクトではなく、引数でジョブ投入スクリプトを渡します。

【sbatch コマンドの例 (スクリプトファイルが run.sh の場合)】

```
$ sbatch run.sh
```

ジョブ投入コマンド (sbatch) について、以下に主なオプションを示します。

表 4.2.1 sbatch オプション

オプション	概要
<code>sbatch -J "sample job name"</code>	ジョブに任意のジョブ名をつけます。
<code>sbatch -p キュー名</code>	指定したキュー名のキュー (パーティション) にジョブを投入します。
<code>sbatch -N ノード数</code>	ノード数を指定します。
<code>sbatch -n プロセス数</code>	ジョブ全体でのプロセス数を指定します。
<code>sbatch -o ./out_%j.log(※)</code>	out_ジョブ ID という名前のファイルに標準出力を出力します。-e オプションが指定されていない場合は、標準エラー出力もこのファイルに出力されます。
<code>sbatch -e ./err_%j.log(※)</code>	err_ジョブ ID という名前のファイルに標準エラー出力を出力します。

※シンボル「%j」はジョブ ID に置換されます。

4.2.2. ジョブ情報表示コマンド (squeue)

ジョブの各種情報を表示するときは、squeue コマンドを実行します。

```
$ squeue
```

注) 他の利用者の情報は表示されません。

【squeue コマンド例】

```
$ squeue
      JOBID PARTITION NAME      USER      ST  TIME  NODES NODELIST(REASON)
      xxxx d024h   testrun  u***0001  R   4:58    60 d[007-066]
      xxxx c006m   testrun  u***0001 CG   0:39     1 c001
```

<出力説明>

JOBID	ジョブに割り当てられたジョブ ID が表示されます。 ジョブの削除(scancel)などで指定する識別子となります。
PARTITION	ジョブを投入したキューの名前を表示します。
NAME	ジョブ名 (未指定の場合はコマンド文字列) を表示します。
USER	ジョブ投入リクエスト (ジョブ) の実行者を表示します。
ST	ジョブの状態を表示します。主なジョブの状態を以下の表に示します。
TIME	ジョブの実行時間 (形式: days-hh:mm:ss)
NODES	ジョブ実行に使用されるノード数
NODELIST (REASON)	ジョブが実行されるホスト名のリスト (ジョブ状態に対する理由があれば表示されます) ジョブ状態が PD (PENDING) の場合の REASON を下記に示します。 (Resources) リソースが利用可能になるのを待っています。 (Priority) パーティション内の優先度の高いジョブが完了するのを待っています。 (Dependency) このジョブが依存する他のジョブが完了するのを待っています。 (InvalidQOS) Resources と同義です。

表 4.2.2 ジョブ状態 (squeue コマンド-ST フィールド)

状態	説明
CA (CANCELLED)	ジョブが明示的にユーザまたはシステム管理者によってキャンセルされました。
CD (COMPLETED)	ジョブは、すべてのノード上のすべてのプロセスを終了しました。
CF (CONFIGURING)	ジョブは、資源が割り当てられた後、資源が使える状態になるのを待っている状態です。
CG (COMPLETING)	ジョブは、終了手続きの過程にあります。
F (FAILED)	ジョブは、ゼロ以外の終了コードまたはその他の障害状態で終了しました。
NF (NODE_FAIL)	ジョブは、割り当てられたノードのいずれかの故障のために終了しました。
PD (PENDING)	ジョブは、資源の割り当てを待っています。保留中です。
PR (PREEMPTED)	ジョブが中断のために終了しました。
R (RUNNING)	ジョブは、現在実行中です。
S (SUSPENDED)	ジョブは、資源の割り当てを持っています (実行が中断されています)。
TO (TIMEOUT)	ジョブは、その制限時間に達して終了しました。

4.2.3. ジョブのキャンセルコマンド (scancel)

ジョブをキャンセルするには `scancel` コマンドを用います。`squeue` コマンドなどで確認したジョブ ID を指定します。`scancel` の後に、ジョブ ID をスペース区切りで指定すると、複数のジョブを一度にキャンセルすることができます。

```
$ scancel ジョブ ID ジョブ ID ジョブ ID
```

4.2.4. ジョブステータス情報表示コマンド (sstat)

ジョブの各種情報（メモリ使用量やディスク I/O 量など）を表示するときは、`sstat` コマンドを実行します。ジョブを実行中に実行する必要があります。

```
$ sstat -j ジョブ ID[.ステップ ID] [-o 項目[,項目]]
```

注) 他の利用者の情報は表示されません。

ジョブが使用中のメモリ量につきましては、下記のコマンドで確認いただけます。

```
$ sstat -j ジョブ ID[.ステップ ID] -o JobID,MaxVMSize,MaxRSS
```

注) 他の利用者の情報は表示されません。

- ステップ ID は `squeue -s` コマンドで確認することができます。
- ステップ ID は省略可能ですが、逐次処理の場合はステップ ID として「batch」を指定します。
- 仮想メモリサイズは `MaxVMSize`、物理メモリ消費サイズは `MaxRSS` を指定します。

ジョブステータス情報表示コマンド (`sstat`) について、以下に主なオプションを示します。

表 4.2.4 sstat オプション

オプション	概要
-j ジョブ ID[.ステップ ID]	ジョブ ID を指定します。必須のオプションです。ステップ ID は省略可能です。ステップ ID はジョブ実行中に <code>squeue -s</code> コマンドで確認することができます。 ※逐次処理の場合はステップ ID として「batch」を指定します。
-o 項目[,項目]...	出力する項目をカンマ区切りで指定します。指定しない場合、全ての項目が表示されます。一部、FOCUS 環境では取得できない項目もあります。 出力項目例 JobID ジョブに割り当てられたジョブ ID が表示されます。 MaxVMSize ジョブの全てのタスクでの仮想メモリサイズの最大値です。 MaxRSS ジョブの全てのタスクでの物理メモリ消費サイズの最大値です。
-e	-o オプションで指定できる項目を表示します。

【sstat コマンド例 1】

```
$ sstat -j 45643 -o JobID,MaxVMSize,MaxRSS,MaxPages,MaxDiskRead,MaxDiskWrite
      JobID  MaxVMSize      MaxRSS MaxPages  MaxDiskRead MaxDiskWrite
```

45643.0	1332496K	39276K	293K	1M	0.29M
---------	----------	--------	------	----	-------

【sstat コマンド例2】

```
$ sstat -j 45638.0
```

JobID	MaxVMSize	MaxVMSizeNode	MaxVMSizeTask	AveVMSize	MaxRSS	MaxRSSNode	MaxRSSTask	AveRSS	
MaxPages	MaxPagesNode	MaxPagesTask	AvePages	MinCPU	MinCPUNode	MinCPUTask	AveCPU	NTasks	AveCPUFreq
ReqCPUFreq	ConsumedEnergy	MaxDiskRead	MaxDiskReadNode	MaxDiskReadTask	AveDiskRead	MaxDiskWrite	MaxDiskWriteNode	MaxDiskWriteTask	AveDiskWrite
45638.0	1579688K	g002	1	912640K	38696K	g002	1	26118K	
279K	g002	1	214K	00:00.000	g001	0	00:00.000	2	2.50G
0	0.96M	g002	1	0.68M	0.02M	g002	1	0.02M	

<出力説明>

JobID	ジョブに割り当てられたジョブ ID が表示されます。
MaxVMSize	ジョブの全てのタスクでの仮想メモリサイズの最大値です。
MaxVMSizeNode	MaxVMSize が発生しているノードです。
MaxVMSizeTask	MaxVMSize が発生しているタスクの ID です。
AveVMSize	ジョブの全てのタスクでの仮想メモリサイズの平均値です。
MaxRSS	ジョブの全てのタスクでの物理メモリ消費サイズの最大値です。
MaxRSSNode	MaxRSS が発生しているノードです。
MaxRSSTask	MaxRSS が発生しているタスクの ID です。
AveRSS	ジョブの全てのタスクの物理メモリ消費サイズの平均値です。
MaxPages	ジョブの全てのタスクでのページフォールト数の最大値です。
MaxPagesNode	MaxPages が発生しているノードです。
MaxPagesTask	MaxPages が発生しているタスクの ID です。
AvePages	ジョブの全てのタスクでのページフォールト数の平均値です。
MinCPU	ジョブの全てのタスクでの CPU 時間の最小値です。
MinCPUNode	MinCPU が発生しているノードです。
MinCPUTask	MinCPU が発生しているタスクの ID です。
AveCPU	ジョブの全てのタスクでの CPU 時間の平均値です。
NTasks	ジョブまたはステップ中でのタスクの合計数です。
AveCPUFreq	ジョブの全てのタスクでの重み付けをした CPU 周波数の平均値 (kHz) です。
ReqCPUFreq	ステップに要求された CPU 周波数 (kHz) です。
ConsumedEnergy	ジョブの全てのタスクでの消費エネルギー (ジュール) です。
MaxDiskRead	ジョブの全てのタスクでの読み込み量の最大値 (bytes) です。
MaxDiskReadNode	MaxDiskRead が発生しているノードです。
MaxDiskReadTask	MaxDiskRead が発生しているタスクの ID です。
AveDiskRead	ジョブの全てのタスクでの読み込み量の平均値 (bytes) です。
MaxDiskWrite	ジョブの全てのタスクでの書き込み量の最大値 (bytes) です。
MaxDiskWriteNode	MaxDiskWrite が発生しているノードです。
MaxDiskWriteTask	MaxDiskWrite が発生しているタスクの ID です。
AveDiskWrite	ジョブの全てのタスクでの書き込み量の平均値 (bytes) です。

4.2.5. 実行ジョブ情報表示コマンド (sacct)

実行が完了したジョブの情報を表示するには sacct コマンドを用います。

```
$ sacct -j ジョブ ID -o 項目[,項目,...] -X
```

注) 他の利用者の情報は表示されません。

実行が完了したジョブのメモリ量につきましては、下記のコマンドで確認いただけます。

```
$ sacct -j ジョブ ID -o JobID,MaxVMSize,MaxRSS
```

注) 他の利用者の情報は表示されません。

sacct コマンドについて、以下に主なオプションを示します。

表 4.2.5 sstat オプション

オプション	概要
-j ジョブ ID	ジョブ ID を指定します。
-o 項目[,項目]...	出力する項目をカンマ区切りで指定します。指定しない場合、 -o 'JobID,JobName,Partition,Account,AllocCPUS,State,ExitCode' を指定した場合と同様の結果となります。 一部、FOCUS 環境では取得できない項目もあります。
-e	-o オプションで指定できる項目を表示します。
-X	ジョブ情報のみ表示し、ステップ毎の情報は表示しません。
-S, --starttime	指定の日時以降の情報を表示します。指定しない場合当日の 0:00 が設定されます。
-E, --endtime	指定の日時以前の情報を表示します。

【sacct コマンド例 1】

```
$ sacct -j xxxxxx -o User,JobID,Partition,NNodes,Submit,Start,End,Elapsed,State -X

      User      JobID Partition  NNodes          Submit          Start          End    Elapsed
State
-----
uxxx000X xxxxxx  c006m    1 2015-11-10T09:24:27 2015-11-10T09:24:31 2015-11-10T09:24:35 00:00:04 COMPLETED
```

<出力説明>

User	ジョブ投入リクエスト（ジョブ）の実行者が表示されます。
JobID	ジョブに割り当てられたジョブ ID が表示されます。
Partition	ジョブを投入したキューの名前が表示されます。
NNodes	ジョブのノード数が表示されます。
Submit	ジョブを投入した日時が表示されます。
Start	ジョブの実行が開始した日時が表示されます。
End	ジョブの実行が完了した日時が表示されます。
Elapsed	ジョブの実行時間（形式：days-hh:mm:ss）
State	ジョブの状態を表示します。

4.3. ジョブの実行方法 (fj コマンド編)

スーパーコンピュータ「京」にて稼働しているジョブ管理コマンド (fj コマンド) も SLURM コマンドと同様に FOCUS スパコン上にて実行できます。

ジョブを実行するには、フロントエンドサーバからコマンドを実行します。

ジョブ管理のためのコマンドは次のとおりです。

表 4.3 ジョブ管理コマンド一覧 (fj コマンド編)

コマンド	コマンド用途	手順
sinfo -s	キュー (パーティション) の情報を表示する	4.1.2. キュー情報の確認方法
squeues	キューのノード実行状況を表示する	
freenodes	空ノード数を表示する	4.1.3. 利用可能なノード数の確認方法
fjsub	ジョブを投入する	4.3.1. ジョブ投入コマンド (fjsub)
fjstat	ジョブやキューの状態・情報を表示する	4.3.2. ジョブ情報表示コマンド (fjstat)
fjdel	ジョブをキャンセルする	4.3.3. ジョブのキャンセルコマンド (fjdel)

※fjsub コマンドを使用するための環境変数はシステム側で自動的に設定します。そのため、利用者側での設定操作は不要です。
(/etc/prpfile.d 配下に wrapper.csh、wrapper.sh を用意しており、自動的にパスが設定されます。)

4.3.1. ジョブ投入コマンド (fjsub)

ジョブを実行するために、ジョブ投入スクリプトを事前に作成します。fjsub コマンドにジョブ投入スクリプトを指定することで、ジョブがキューイングされ実行されます。

【fjsub コマンドの書式】

```
$ fjsub ジョブ投入スクリプト
```

注) bsub と異なり、リダイレクトではなく、引数でジョブ投入スクリプトを渡します。

【fjsub コマンドの例 (スクリプトファイルが run.sh の場合)】

```
$ fjsub run.sh
```

ジョブ投入コマンド (fjsub) について、以下に主なオプションを示します。

表 4.3.1 fjsub オプション

オプション	概要
fjsub -N "sample job name"	ジョブに任意のジョブ名をつけます。
fjsub -L node=ノード数	ノード数を指定します。
fjsub -L elapse=経過時間	ジョブの経過時間制限値を指定します。 経過時間は分単位で指定することができます。
fjsub -o ./out_%j.log(※)	out_ジョブ ID という名前のファイルに標準出力を出力します。-e オプションが指定されていない場合は、標準エラー出力もこのファイルに出力されます。
fjsub -e ./err_%j.log(※)	err_ジョブ ID という名前のファイルに標準エラー出力を出力します。

※シンボル「%j」はジョブ ID に置換されます。

4.3.2. ジョブ情報表示コマンド (fjstat)

ジョブの各種情報を表示するときは、fjstat コマンドを実行します。fjstat の後に、ジョブ ID をスペース区切りで複数指定すると、複数のジョブを一度に表示することができます。また、ジョブ ID を省略すると、参照可能な全てのジョブ情報を表示することができます。

```
$ fjstat ジョブ ID
$ fjstat ジョブ ID ジョブ ID ジョブ ID
$ fjstat
```

注) 他の利用者の情報は表示されません。

【fjstat コマンド例】

```
$ fjstat
      QUEUED      RUNNING      HOLD      ERROR      TOTAL
      0           3           0           0           3
s      0           3           0           0           3

JOB_ID      JOB_NAME MD  ST      USER      START_DATE  ELAPSE_LIM  NODE_REQUIRE
  1 test_test_test NM  RUN      *****  12/27 11:34:39 0001:40:00      1
  2 test_test_test NM  RUN      *****  12/27 11:49:18 0000:06:00      1
  3 test_test_test NM  RUN      *****  12/27 11:55:00 0001:40:00      1
```

表 4.3.2-1 fjstat オプション

オプション	概要
[-c <clustername>]	指定したクラスタに登録されたジョブを集計対象とします。
[-A --all]	全てのクラスタに登録されたジョブを集計対象とします。
[--choose <item> [, <item> ...]]	集計情報を表示する際の表示項目とその順序 (位置) を指定します。
[<jobid> [<jobid> ...]]	集計対象のジョブ ID を指定します。
--help	本コマンドの使用方法を表示します。

表 4.3.2-2 ジョブステータス

ステータス	略称	意味
QUEUE	QUE	ジョブ実行待ち状態のジョブ件数
RUNNING	RUN	ジョブ実行中のジョブ件数
HOLD	HLD	ユーザにより停止されたジョブ件数
ERROR	ERR	エラーにより停止されたジョブ件数
—	EXT	ジョブ終了処理完了
—	CCL	ジョブ実行中止による終了

表 4.3.2-3 表示アイテム

アイテム名	説明	詳細
JOB_ID	ジョブ ID	ジョブ登録時に SLURM が発行するジョブ ID (整数)
JOB_NAME	ジョブ名	ジョブ登録時にユーザが指定したジョブ名 (デフォルトではスクリプトファイル名)
MD	ジョブモデル	常に NM の 2 文字固定
ST	ジョブステータス	ジョブのステータス名 (表 5.1.2.3-2 ジョブステータス)
USER	実行ユーザ名	アカウント名
START_DATE	ジョブ開始時刻	ジョブ実行前の場合は、開始予測時間“YYYY/MM/DD”を出力 実行中および実行後の場合は、実際に開始した時刻“MM/DD hh:mm:ss”を出力 (予測時刻)

		の場合は時刻が括弧で囲まれて出力される)
ELAPSE_LIM	経過時間制限	"hhhh:mm:ss"の形式 桁が溢れる場合は、ss を省略して出力
NODE_REQUIRE	ジョブの投入時のノード数	ノード数

4.3.3. ジョブのキャンセルコマンド (fjdel)

ジョブをキャンセルするには fjdel コマンドを用います。fjstat コマンドなどで確認したジョブ ID を指定します。fjdel の後に、ジョブ ID をスペース区切りで指定すると、複数のジョブを一度にキャンセルすることができます。

```
$ fjdel ジョブ ID ジョブ ID ジョブ ID
```

4.4. ジョブ投入スクリプトの作成

ジョブ投入のためのジョブ投入スクリプトを作成します。

4.4.1. 処理方法の指定

ジョブ投入スクリプトの中で“#SBATCH”で始まる行に sbatch オプションを記述すると、処理方法を指定することができます。

4.4.2. sbatch オプション

主な sbatch オプションを次に示します。

表 4.4.2 sbatch オプション

ジョブ投入スクリプトのディレクティブ	#SBATCH
キュー（パーティション）指定	-p [queue]
実行ノード数（並列数の指定）	-N [minnodes[-maxnodes]], --nodes=[minnodes[-maxnodes]]
CPU 数の指定（プロセス数の指定）	-n [number], --ntasks=[number]
実行時間の上限指定（wall time）	-t [minutes], --time=[minutes] -t [minutes:seconds], --time=[minutes:seconds] -t [hours:minutes:seconds], --time=[hours:minutes:seconds] -t [days-hours], --time=[days-hours] -t [days-hours:minutes], --time=[days-hours:minutes] -t [days-hours:minutes:seconds], --time=[days-hours:minutes:seconds]
出力ファイル指定	-o [filename], --output=[filename]
エラー出力指定	-e [filename], --error=[filename]
出力・エラー出力の総合出力	(-e 指定無しで-o を使用)
イベント通知	--mail-type=[type] ※type は BEGIN, END, FAIL, REQUEUE, ALL のいずれか
メールアドレス指定	--mail-user=[address]
ジョブ再投入	--requeue または --no-requeue (未指定時は--no-requeue)
実行ディレクトリ指定	--workdir=[dir_name]
メモリサイズ指定	--mem=[mem] [M G T] OR --mem-per-cpu=[mem] [M G T]
ノード当たりのタスク数指定	--tasks-per-node=[count]
タスク当たりの CPU 数指定	--cpus-per-task=[count]
依存ジョブ	--dependency=[type:job_id] type には次の依存タイプを指定できます。 after 指定したジョブが開始された後に、このジョブが実行されます。 afterany 指定したジョブが終了した後に、このジョブが実行されます。 afternotok 指定したジョブが異常終了した後に、このジョブが実行されます。 afterok 指定したジョブが正常終了した後に、このジョブが実行されます。 同じ依存タイプに複数のジョブ ID を紐付ける場合は、コロン (:) で区切ります。 #SBATCH --dependency=type:job_id: job_id 複数の依存関係（依存タイプとジョブ ID の組み合わせ）を指定する場合は、依存関係の間をカンマ (,) で区切ります。 #SBATCH --dependency=type:job_id,type:job_id

ジョブのプロジェクト化	--wckey=[name]
ジョブ実行ホストの詳細	--odelist=[nodes] AND/OR --exclude=[nodes]
アレイジョブ	--array=[array_spec]
開始時間指定	--begin=YYYY-MM-DD[THH:MM[:SS]]

4.4.3. 環境変数

sbatch コマンド実行時、ジョブ実行時に環境変数が設定されます。設定される主な環境変数を示します。

表 4.4.3 環境変数

環境変数	内容
SLURM_JOB_CPUS_PER_NODE	ジョブ実行に使用されるホストとプロセス数のリスト
SLURM_JOB_ID	ジョブ ID
SLURM_JOB_NAME	fjsub -N や sbatch -J で指定したジョブ名。ジョブ名を指定していない場合は、実際に指定されたコマンド列が格納されます。
SLURM_JOB_NODELIST	ジョブが実行されるホスト名のリスト
SLURM_NTASKS	sbatch -n(または --ntasks) で指定したプロセス数
SLURM_SUBMIT_DIR	ジョブが投入されたカレントディレクトリ

4.4.4. 逐次ジョブを実行する場合

逐次（並列計算を行わない方式）で実行する時に作成するジョブ投入スクリプトの例を示します。

```
#!/bin/bash
#SBATCH -p d006m                ..... (1)
#SBATCH -n 1                    ..... (2)
#SBATCH -J test_serial          ..... (3)
#SBATCH -o stdout.%J            ..... (4)
#SBATCH -e stderr.%J           ..... (5)

./a.out                        ..... (6)
```

- (1) パーティション名（キュー名）を指定します。
- (2) ジョブで使用するプロセス数（-n もしくは--ntasks=<number>）を指定します。
逐次ジョブの場合、プロセス数は1 となりますので、指定値は“-n 1”を指定します。
- (3) ジョブ名を指定します。
- (4) 標準出力ファイルを指定します。％J はジョブ ID に変換されます。
- (5) 標準エラー出力ファイルを指定します。％J はジョブ ID に変換されます。
- (6) プログラム（a.out）を実行します。

4.4.5. スレッド並列ジョブを実行する場合

スレッド並列（単体ノードで並列計算を行う方式）でジョブを実行する時に作成するジョブ投入スクリプトの例を示します。

```
#!/bin/bash
#SBATCH -p d006m ..... (1)
#SBATCH -n 1 ..... (2)
#SBATCH -c 20 ..... (3)
#SBATCH -J test_omp ..... (4)
#SBATCH -o stdout.%J ..... (5)
#SBATCH -e stderr.%J ..... (6)

export OMP_NUM_THREADS=${SLURM_CPUS_PER_TASK} ..... (7)
./a.out ..... (8)
RETCODE=$?
exit ${RETCODE} ..... (9)
```

- (1) パーティション名（キュー名）を指定します。
- (2) ジョブで使用するプロセス数（`-n` もしくは `--ntasks=<number>`）を指定します。
上記の例では、1 プロセスで実行するために `-n 1` を指定しています。
- (3) ジョブで使用する CPU 数（`-c` もしくは `--cpus-per-task=<number>`）を指定します。
上記の例では、20 スレッドで実行するために `-c 20` を指定しています。
- (4) ジョブ名を指定します。
- (5) 標準出力ファイルを指定します。`%J` はジョブ ID に変換されます。
- (6) 標準エラー出力ファイルを指定します。`%J` はジョブ ID に変換されます。
- (7) 環境変数 `OMP_NUM_THREADS` にスレッド数を指定します。
- (8) プログラム（`a.out`）を実行します。
- (9) プログラムの戻り値を `SLURM` に戻します。

4.4.6. MPI プログラム (OpenMPI) を実行する場合

プロセス並列 (複数のノードで並列計算を行う方式) のジョブを OpenMPI で実行する時に作成するジョブ投入スクリプトの例を示します。

下記スクリプトでは、D システムにて 2 ノード (1 ノードあたり 2 プロセス) のジョブが生成されます。

```
#!/bin/bash
#SBATCH -p d006m ..... (1)
#SBATCH -n 40 ..... (2)
#SBATCH -J test_openmpi ..... (3)
#SBATCH -o stdout.%J.log ..... (4)
#SBATCH -e stderr.%J.log ..... (5)

module load MPI-openmpi-2.1.3+gnu-4.4.7 ..... (6)
mpirun -np ${SLURM_NTASKS} ./a.out ..... (7)
RETCODE=$?
exit ${RETCODE} ..... (8)
```

- (1) パーティション名 (キュー名) を指定します。
- (2) ジョブで使用するプロセス数 (-n もしくは--ntasks=<number>) を指定します。
- (3) ジョブ名を指定します。
- (4) 標準出力ファイルを指定します。 %J はジョブ ID に変換されます。
- (5) 標準エラー出力ファイルを指定します。 %J はジョブ ID に変換されます。
- (6) MPI 環境変数をセットします。
- (7) プログラム (a.out) を実行します。
- (8) プログラムの戻り値を SLURM に戻します。

上記 (6) については、次の実行方法があります。

標準 GNU コンパイラー (GNU 4.4.7)	OpenMPI	module load MPI-openmpi-2.1.3+gnu-4.4.7
GNU6.3.0 コンパイラー	OpenMPI	module load PrgEnv-gnu-6.3.0 module load MPI-openmpi-2.1.1+gnu-6.3.0
Intel コンパイラー	OpenMPI	module load PrgEnv-intel-17.0.1.132 module load MPI-openmpi-2.1.3+intel-17.0.1.132
	Intel MPI	module load PrgEnv-intel-18.0.3.222 module load MPI-impi-18.3.222

4.4.7. MPI プログラム (Intel MPI) を実行する場合

プロセス並列 (複数のノードで並列計算を行う方式) のジョブを Intel MPI で実行する時に作成するジョブ投入スクリプトの例を示します。

```
#!/bin/bash
#SBATCH -p d006m
#SBATCH -n 40
#SBATCH -J test_intelmpi
#SBATCH -o stdout.%J.log
#SBATCH -e stderr.%J.log

module load PrgEnv-intel-18.0.3.222
module load MPI-impi-18.3.222
NODEFILE=`generate_pbs_nodefile`
mpirun -np ${SLURM_NTASKS} ./a.out
```

(注) インテルコンパイラで作成された実行ファイルを実行する場合、ジョブ投入スクリプト内に
module load PrgEnv-intel-18.0.3.222
module load MPI-impi-18.3.222
の行を記載する必要があります。

4.4.8. MPI プログラム (mpich2) を実行する場合

現在 MPICH2 環境の module は提供していません。

4.5. Xeon Phi コプロセッサの使用

演算ノード側の CPU にプログラムとデータを持ち、必要な時に処理対象のプログラムとデータのみを Xeon Phi に引き渡しその処理結果を得ることができます。

MKL (Math Kernel Library) を使っているプログラムでは、その計算の一部を Xeon Phi に自動でオフロードする機能が提供されています。

FOCUS スパコンは通常、演算ノードへのログインは許可していません。

したがってフロントエンドサーバ ff[01-04]や E システムの演算ノードに備えられた Xeon Phi も同様に、ログインせずに利用するオフロードモードでの利用に限ります。

4.6. 課金確認コマンド

4.6.1. プロジェクト単位従量課金確認コマンド `thismonth`

基本サービスでの本年度当月初めからの終了ジョブに対する従量課金額等の情報が得られます。
`thismonth` (引数無し) で当月の情報が、`thismonth YYYYMM` で西暦と月を与えると該当年月の情報が得られます。

指定年度の情報を表示する場合は `thismonth_HYY` を実行します。 ※HYY は任意年度

`thismonth_HYY YYYYMM` で西暦と月を与えることで、指定年度の該当年月の情報が得られます。

【実行例1】

```
$ thismonth
Charge information of *** in this month.
system  njob    avg_et avg_nodes avg_procs et_nodes  charge  et_max et_np_max  np_max
A         23      0.4    15.3    184.5    19.4    1725.0   8.3    166.7    924
B         16      0.0     1.6     26.2     0.5     37.0    0.1     4.0     32
D         49      0.1    18.3    365.7    48.8   13078.0  0.2    77.6   1600
E         22      0.0    10.4    208.2     3.3     994.0   0.3    47.8    620
F          0      0.0     0.0     0.0     0.0      0.0    0.0     0.0     0
G          1      0.0     1.0    20.0     0.0      6.0    0.0     0.7     20
H          0      0.0     0.0     0.0     0.0      0.0    0.0     0.0     0
V         12      0.0     3.0    36.7     0.3     19.0    0.0     0.4     60

TOTAL: 15859 yen
```

【実行例2】

```
$ thismonth 201904
Charge information of *** in this month.
system  njob    avg_et avg_nodes avg_procs et_nodes  charge  et_max et_np_max  np_max
A         23      0.4    15.3    184.5    19.4    1725.0   8.3    166.7    924
B         16      0.0     1.6     26.2     0.5     37.0    0.1     4.0     32
D         49      0.1    18.3    365.7    48.8   13078.0  0.2    77.6   1600
E         22      0.0    10.4    208.2     3.3     994.0   0.3    47.8    620
F          0      0.0     0.0     0.0     0.0      0.0    0.0     0.0     0
G          1      0.0     1.0    20.0     0.0      6.0    0.0     0.7     20
H          0      0.0     0.0     0.0     0.0      0.0    0.0     0.0     0
V         12      0.0     3.0    36.7     0.3     19.0    0.0     0.4     60

TOTAL: 15859 yen
```

【実行例3】

```
$ thismonth_H30 201804
Charge information of xxx in this month.
system  njob    avg_et avg_nodes avg_procs et_nodes  charge  et_max et_np_max  np_max
A         257      0.1     1.6    19.2    34.2   3078.0   21.0   251.7    240
B         240      0.0     1.1    16.8     4.4   251.0    0.1     4.7     32
C         355      0.0     1.3    15.3    10.4   661.0    0.2    48.5    216
D         244      0.0     1.4    27.9     9.3  2254.0    0.3   100.6    400
E         250      0.0     1.4    27.5     8.0  2083.0    0.2    70.3    320
F         465      0.0     1.1    45.8     8.1  3970.0    0.0     0.8    400
G           0      0.0     0.0     0.0     0.0    0.0     0.0     0.0     0
H         257      1.1     1.5    12.8   292.3 29037.0   72.0   576.1    240
V           0      0.0     0.0     0.0     0.0    0.0     0.0     0.0     0

TOTAL: 41334 yen
```

<出力説明>

system	システム名
njob	ジョブ数
avg_et	平均経過時間 (h)
avg_nodes	平均ノード数
avg_procs	平均プロセス数
et_nodes	ノード時間 (h)
charge	課金額 (円) ※H26 年度の FOCUS 賛助会員の価格はカッコ付きで表示
et_max	最大経過時間 (h)
et_np_max	最大プロセスノード時間 (h)
np_max	最大プロセス数

<注意>

コマンドの出力結果に下記の項目は反映されておりません。

- ・実行中のジョブおよび当日終了ジョブ分
(コマンド出力への反映は、ジョブ実行終了日の 24:00 です。 参考・「4.6.4. 課金利用コマンドの情報反映タイミング」)
- ・トライアルコース、FOCUS 賛助会員加入特典等の割引額
- ・期間占有利用 (計算資源予約) の利用分
- ・連休特別キュー (GW 特別キュー等) の利用分

4.6.2. 利用者単位従量課金確認コマンド **uacct**

基本サービスでの本年度の終了ジョブに対する従量課金額等の情報が得られます。

uacct (引数無し) で当月の情報が、uacct YYYYMM で西暦と月を与えると該当年月の情報が得られます。

指定年度の情報を表示する場合はuacct_HYY を実行します。 ※HYY は任意年度

uacct_HYY YYYYMM で西暦と月を与えることで、指定年度の該当年月の情報が得られます。

【実行例1】

```
$ uacct
Charge information of ***** in 201904 :
Computational: 945 (Rack rate: 945)
Items
A:      9 nodehours, 833 yen
B:      1 nodehours, 3 yen
D:      1 nodehours, 20 yen
E:      1 nodehours, 83 yen
F:      0 nodehours, 0 yen
G:      1 nodehours, 6 yen
H:      0 nodehours, 0 yen
V:      0 nodehours, 0 yen
=====
Total charge : 945
```

【実行例2】

```
$ uacct 201904
Charge information of ***** in 201904 :
Computational: 945 (Rack rate: 945)
Items
A:      9 nodehours, 833 yen
B:      1 nodehours, 3 yen
D:      1 nodehours, 20 yen
E:      1 nodehours, 83 yen
F:      0 nodehours, 0 yen
G:      1 nodehours, 6 yen
H:      0 nodehours, 0 yen
V:      0 nodehours, 0 yen
=====
Total charge : 945
```

【実行例 3】

```
$ uacct_H30 201804
Charge information of afse0025 in 201804 :
Computational: 7842 (Rack rate: 7842)
Items
A:      9 nodehours, 641 yen
B:      1 nodehours, 152 yen
C:      1 nodehours, 89 yen
D:     17 nodehours, 5154 yen
E:       0 nodehours, 1806 yen
F:       0 nodehours,  0 yen
G:      1 nodehours,  6 yen
H:       0 nodehours,  0 yen
V:       0 nodehours,  0 yen

=====
Total charge : 7842
```

<出力説明>

Computational:	演算ノード課金額
Rack rate:	演算ノード割引前課金
Items	内訳
A :	A システム ノード時間、課金額
B :	B システム ノード時間、課金額
C :	C システム ノード時間、課金額
D :	D システム ノード時間、課金額
E :	E システム ノード時間、課金額 ※H26 年度の FOCUS 賛助会員の価格はカッコ付きで表示
F :	F システム ノード時間、課金額
G :	G システム ノード時間、課金額
H :	H システム ノード時間、課金額
V :	V システム ノード時間、課金額
Total charge	合計課金額

<注意>

コマンドの出力結果に下記の項目は反映されておりません。

- ・実行中のジョブおよび当日終了ジョブ分
(コマンド出力への反映は、ジョブ実行終了日の 24:00 です。 参考・「4.6.4. 課金利用コマンドの情報反映タイミング」)
- ・トライアルユース、FOCUS 賛助会員加入特典等の割引額
- ・期間占有利用 (計算資源予約) の利用分
- ・連休特別キュー (GW 特別キュー等) の利用分

4.6.3. 利用者単位アプリケーション課金確認コマンド `uacct_apl`

Gaussian, MIZUHO/BioStation の従量課金額等の情報が得られます。

`uacct_apl` (引数無し) で当月の情報が、`uacct_apl YYYYMM` で西暦と月を与えると該当年月の情報が得られます。

指定年度の情報を表示する場合は `uacct_apl_HYY` を実行します。 ※HYY は任意年度

`uacct_apl_HYY YYYYMM` で西暦と月を与えることで、指定年度の該当年月の情報が得られます。

【実行例1】

```
$ uacct_apl
Charge information of ***** in 201904 :
Computational: 245 (Rack rate: 245)
  Items
      Gaussian:      1 nodehours, 19 yen
  MIZUHO/BioStation: 2 nodehours, 133 yen
=====
Total charge : 152
```

【実行例2】

```
$ uacct_apl 201904
Charge information of ***** in 201904 :
Computational: 245 (Rack rate: 245)
  Items
      Gaussian:      1 nodehours, 19 yen
  MIZUHO/BioStation: 2 nodehours, 133 yen
=====
Total charge : 152
```

【実行例3】

```
$ uacct_apl_H30 201804
Charge information of ***** in 201804 :
Computational: 54 (Rack rate: 54)
  Items
      Gaussian:      0 nodehours, 51 yen
  MIZUHO/BioStation: 0 nodehours, 1 yen
=====
Total charge : 54
```

<出力説明>

Computational:	課金額
Rack rate:	割引前課金
Items	内訳
Gaussian:	Gaussian 09,16 ノード時間、課金額
Gaussian 09:	Gaussian 09 ノード時間、課金額(2016 年度以前)
MIZUHO/BioStation:	MIZUHO/BioStation ノード時間、課金額
Parallel CONFLEX:	Parallel CONFLEX ノード時間、課金額(2016 年度以前)
Total charge	合計課金額

<注意>

デバッグキューや専用キューにて実行されたジョブについてもアプリケーション利用料金が発生いたします。

また、コマンドの出力結果に下記の項目は反映されておりません。

- ・実行中のジョブおよび当日終了ジョブ分

(コマンド出力への反映は、ジョブ実行終了日の 24:00 です。 参考・「4.6.4. 課金利用コマンドの情報反映タイミング」)

4.6.4. 課金確認コマンドの情報反映タイミング

課金確認コマンドは、ジョブ実行履歴データを利用して課金計算を行っています。

ジョブ実行履歴データは毎日 0 時の時点で終了しているジョブを反映します。

その為、課金コマンドを実行するタイミングにより課金計算に含まれないジョブが存在する場合があります。

(例 1) 7 月 9 日 14:00 に終了したジョブは、7 月 10 日 0:00 以降に課金確認コマンドに反映されます。

(例 2) 7 月 10 日 0:00 時点で実行中のジョブは 7 月 10 日の課金確認コマンドには未反映となります。

4.7. 実行ジョブ一覧の確認方法

実行したジョブの一覧を確認するには2つの方法があります。

- (1) sacct コマンド (4.2.5.実行ジョブ情報表示コマンド (sacct)) にて確認
 sacct コマンドにて -S オプションを使って指定の日時以降の情報を表示することができます。
 オプションの詳細については「4.2.5.実行ジョブ情報表示コマンド (sacct)」をご覧ください。

【sacct コマンド例】(2019.4.01 以降のジョブを表示)

```
$ sacct -S 2019-04-01 -o User,JobID,Partition,NNodes,Submit,Start,End,Elapsed,State -X
```

User	JobID	Partition	NNodes	Submit	Start	End	Elapsed	State
uxxx000X	xxxxxxx	a006m	1	2019-04-10T09:24:27	2019-04-10T09:24:31	2019-04-10T09:24:35	00:00:04	COMPLETED
uxxx000X	yyyyyy	g006m	1	2019-04-12T16:07:11	2019-04-12T16:07:14	2019-04-12T16:08:36	00:01:22	COMPLETED
uxxx000X	zzzzzz	f006m	1	2019-04-15T10:36:45	2019-04-15T10:36:49	2019-04-15T10:36:59	00:00:10	COMPLETED

- (2) AccountingLog からの確認

実行したジョブの情報は

/home1/[利用グループ]/AccountingLog に保存されています。

cat コマンド等で表示してご確認ください。

ログファイルは日別・ユーザ別・システム別(ABC, DEG または FH)に出力されます。

(ファイル名フォーマット YYYYMMDD_uxxx000X_ABC_acct.log,
 YYYYMMDD_uxxx000X_DEG_acct.log, YYYYMMDD_uxxx000X_FH_acct.log)

情報が反映されるのは毎日 0 時に一度のみです。

(例1) 7 月 9 日 14:00 に終了したジョブは、7 月 10 日 0:00 以降に課金確認コマンドに反映されます。

(例2) 7 月 10 日 0:00 時点で実行中のジョブは 7 月 10 日の課金確認コマンドには未反映となります。

【実行例1】

```
$ cat /home1/gxxx/AccountingLog/20190421_uxxx0001_ABC_acct.log
```

JobId	YYYYMMDDhhmmss	WaitTime	ElapsedTime	NumNodes	NumProcs	UserName	Queue
1212949	20190418130449	1	63	1	12	uxxx0001	a024h
1212953	20190418133617	1	3	1	12	uxxx0001	a024h
1212954	20190418133934	1	6	2	24	uxxx0001	a024h
1212955	20190418134205	0	6	2	24	uxxx0001	a024h

【実行例2】

```
$ cat /home1/gxxx/AccountingLog/20190417_uxxx0001_FH_acct.log
```

JobId	YYYYMMDDhhmmss	WaitTime	ElapsedTime	NumNodes	NumProcs	UserName	Queue
1209018	20190405114113	1	23	1	40	uxxx0001	f024h
1209019	20190405114121	1	22	1	40	uxxx0001	f072h
1209020	20190405114154	1	38	1	40	uxxx0001	f072h_p100
1209024	20190405114305	0	27	1	40	uxxx0001	f006m

<ファイル内容説明>

JobId : ジョブ ID
YYYYMMDDhhmmss : ジョブ完了日時
WaitTime : CPU 待ち時間 (秒)
ElapsedTime : 実行時間 (秒)
NumNodes : ノード数
NumProcs : プロセス数
UserName : ユーザーアカウント
Queue : 利用キュー名

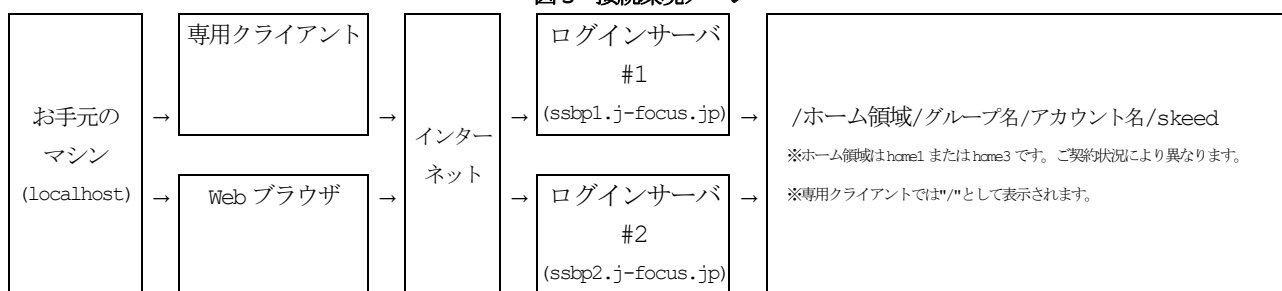
5. インターネット高速転送システムの使用方法

TCP/IP を使った SCP や SFTP より効率の良い Skeed Silver Bullet を導入し、遅延の大きな遠隔地と FOCUS スパコンシステム間で高速なデータ転送を行います。以下の環境のいずれかを利用できます。

表5 クライアント環境

環境	制限	備考
専用クライアント (SkeedSilverBullet GUI)	容量制限無し	専用のソフトウェアが必要です。センターから専用のソフトウェアを貸与します。 ソフトウェアが対応するクライアント OS は Windows と MacOS X です。
Web ブラウザベース	容量制限無し	お手元の Web ブラウザを使います。使用する Web ブラウザは Chrome、Internet Explorer、Firefox が利用可能です。 Web ブラウザから専用クライアントを起動することも可能です。

図5 接続環境イメージ



5.1. Skeed Silver Bullet の利用申請

Skeed Silver Bullet をご利用の際は、「FOCUS スパコン利用ポータルサイト」より予約申請を行ってください。

(<https://portal.j-focus.jp/focus/app/common/reservation/list> ※SSL-VPN 接続が必要)

予約申請は利用開始日の2 業務日前 9 時までをお願いします。

予約状況につきましては、『インターネット高速ファイル転送サーバ (ssb) 予約状況カレンダー』でご確認いただけます。

(<http://www.j-focus.jp/cal/ssb.html>)

5.2. 専用クライアント (SkeedSilverBullet GUI) の使用方法

5.2.1. クライアントのインストール

- (1) 次のいずれかの方法でインストールプログラムを入手します。

(方法 1)

次の手順を参照し、お手元のマシンで SCP 転送の画面を起動します。

- ・『[2.1.1.4. SCP ファイル転送](#)』
- ・『[2.1.2.2. SSL-VPN 接続による SCP ファイル転送 \(WinSCP 使用\)](#)』

フロントエンドサーバの /home1/share/skeed ディレクトリから、次のファイルをお手元のマシンにダウンロードします。

【Windows の場合】

SetupSkeedSilverBulletClient_2.4.1.1.exe

【MacOS X の場合】

SetupSkeedSilverBullet_2.4.1.1.dmg (※)

※本手順では、MacOS X のアプリケーションフォルダに保存することを前提に説明します。

(方法 2)

FOCUS スパコン利用ポータルサイトのクライアントソフトウェアダウンロードからインストールプログラムをダウンロードできます。下記の URL からアクセスしてください。

〔URL〕 <https://portal.j-focus.jp/focus/app/> (SSL-VPN 接続が必要)

- (2) プログラムを起動します。

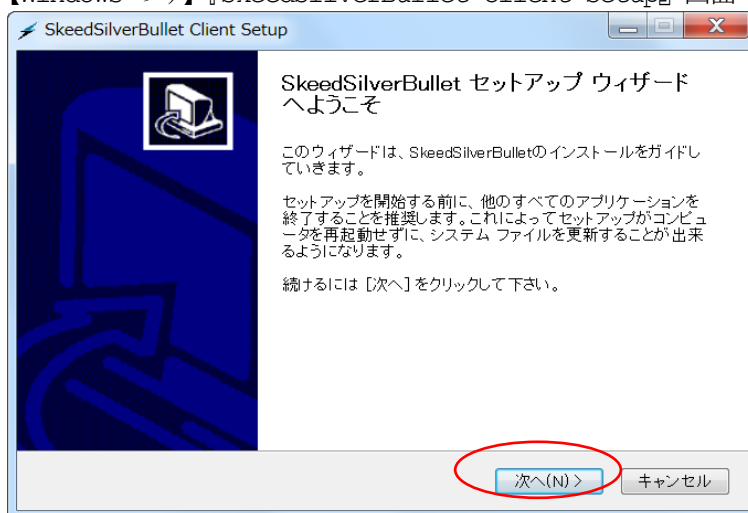
【Windows の場合】

[SetupSkeedSilverBulletClient_2.4.1.1.exe] をダブルクリックします。

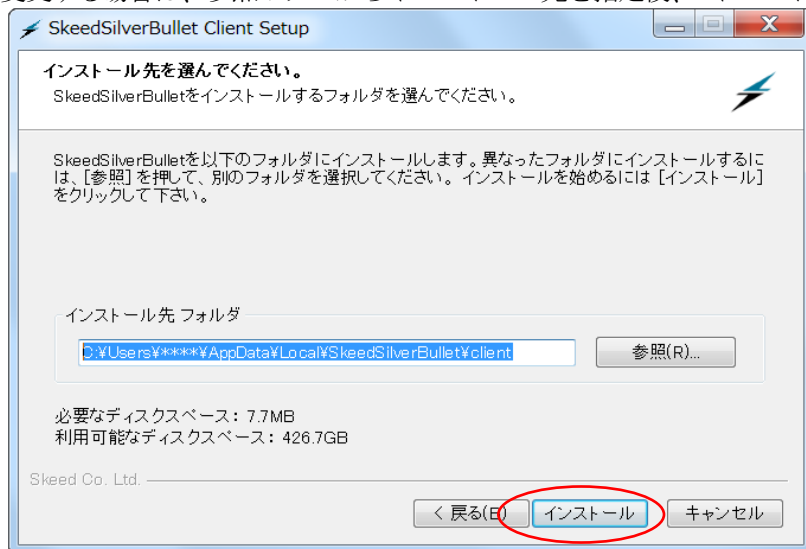
【MacOS X の場合】

[SetupSkeedSilverBullet_2.4.1.1.dmg] をダブルクリックし、展開された [SkeedSilverBullet] をダブルクリックします。

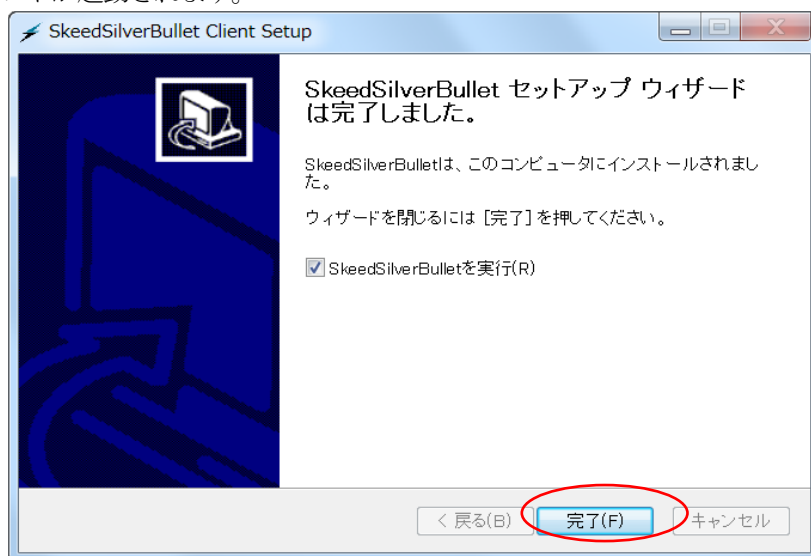
- (3) 【Windows のみ】『SkeedSilverBullet Client Setup』画面で [次へ] ボタンをクリックします。



- (4) 【Windows のみ】インストール先を確認（または指定）し、[インストール] ボタンをクリックします。
※初期値は C:\¥Users¥<アカウント名>\¥AppData¥Local¥SkeedSilverBullet¥client です。
変更する場合は、参照ボタンからインストール先を指定後、「インストール」をクリックします。



- (5) 【Windows のみ】「完了しました」が表示されたことを確認し、[完了] ボタンをクリックします。
※「SkeedSilverBullet を実行」にチェックが入っている場合は SkeedSilverBullet GUI クライアントが起動されます。



※JAVA のインストール

お使いの PC に JAVA がインストールされていない場合には以下の様なダイアログが表示されます。
手順にしたがって JAVA をインストールします。



5.2.2. クライアントの環境設定

- (1) SkeedSilverBullet を起動します。
※SSL-VPN 接続中の場合は切断してください

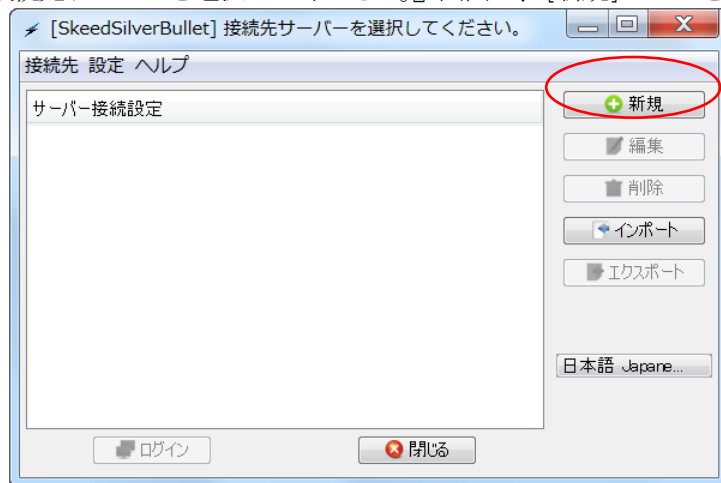
【Windows の場合】

[スタート] → [すべてのプログラム] → [Skeed] → [SilverBulletClient]
→ [SkeedSilverBulletClient] を順に選択します。

【MacOS X の場合】

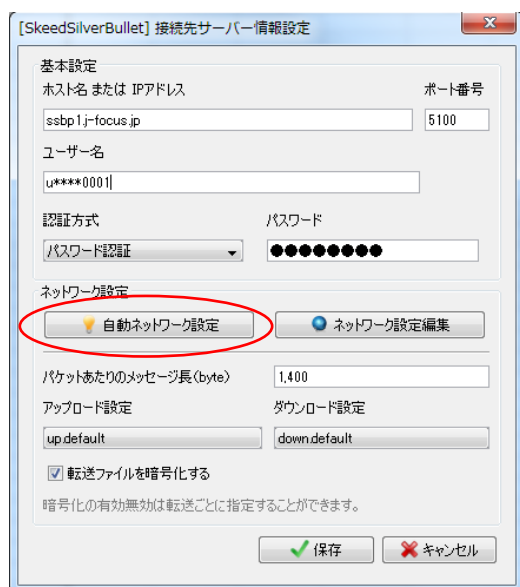
メニュー [移動] → [アプリケーション] → [SkeedSilverBullet] を順に選択します。

- (2) 『接続先サーバーを選択してください。』画面で、[新規] ボタンをクリックします。

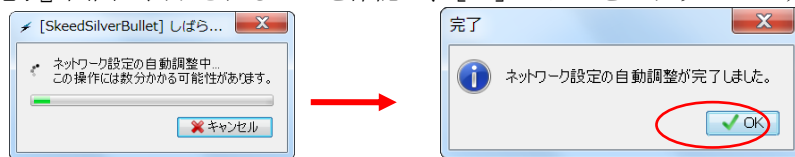


- (3) 次の項目を入力し、画面中央部にある [自動ネットワーク設定] ボタンをクリックします。
接続先サーバは2 つありますので、どちらかに接続してください。

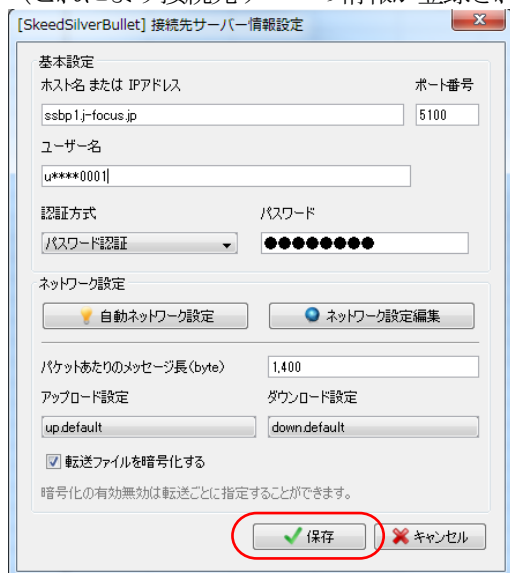
ホスト名	ssbp1.j-focus.jp または ssbp2.j-focus.jp
ポート番号	5100
ユーザー名	アカウント名 (「u」 + 「課題名」 + 数字 4 桁)
認証方式	パスワード認証
パスワード	アカウントのパスワード (変更は『2.2.1.パスワードの変更 (センター内)』を参照)



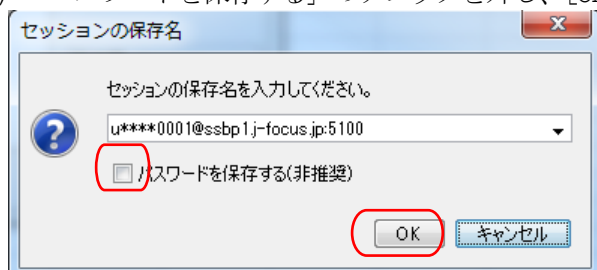
- (4) 『完了』画面が表示されることを確認し、[OK] ボタンをクリックします。



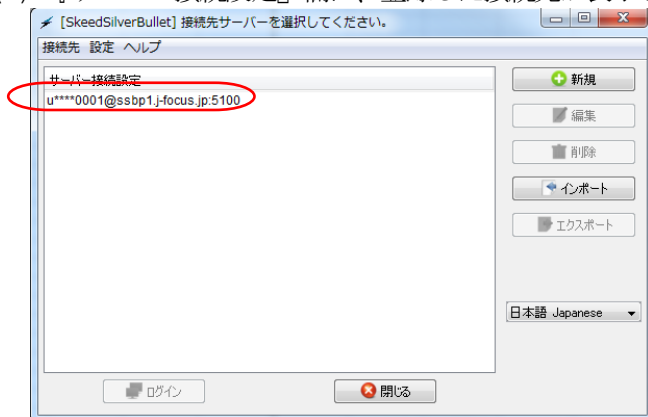
- (5) 画面下部の[保存] ボタンをクリックします。
(これにより接続先サーバの情報が登録され、一覧画面での選択が可能になります。)



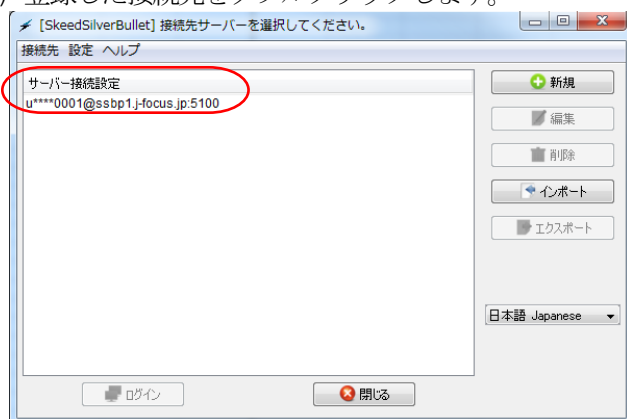
- (6) 「パスワードを保存する」のチェックを外し、[OK] ボタンをクリックします。



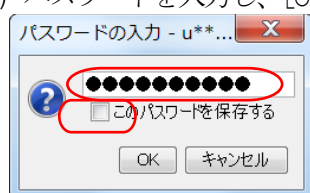
- (7) 『サーバー接続設定』欄に、登録した接続先が表示されることを確認します。



- (8) 登録した接続先をダブルクリックします。



- (9) パスワードを入力し、[OK] ボタンをクリックします。



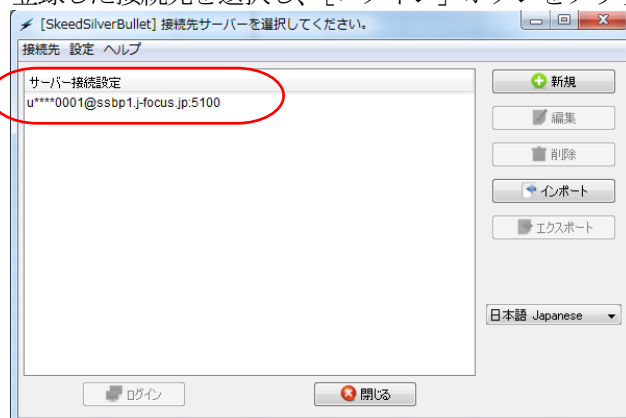
5.2.3. クライアントの起動

- (1) SkeedSilverBullet を起動します。
※SSL-VPN 接続中の場合は切断してください

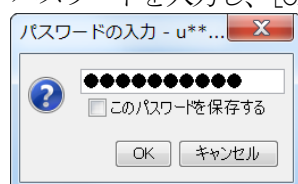
【Windows の場合】 [スタート]→[すべてのプログラム]→[Skeed]→[SilverBulletClient]
→[SkeedSilverBulletClient]を順に選択します。

【MacOS X の場合】
メニュー[移動]→[アプリケーション]→[SkeedSilverBullet]を順に選択します。

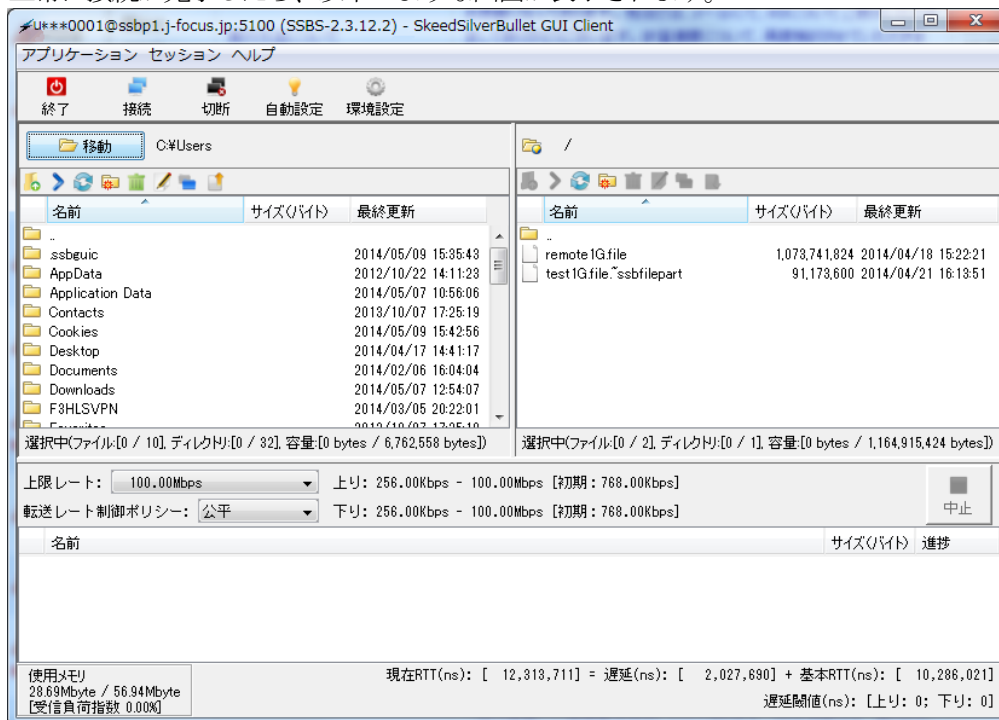
- (2) 登録した接続先を選択し、[ログイン] ボタンをクリックします。



- (3) パスワードを入力し、[OK] ボタンをクリックします。

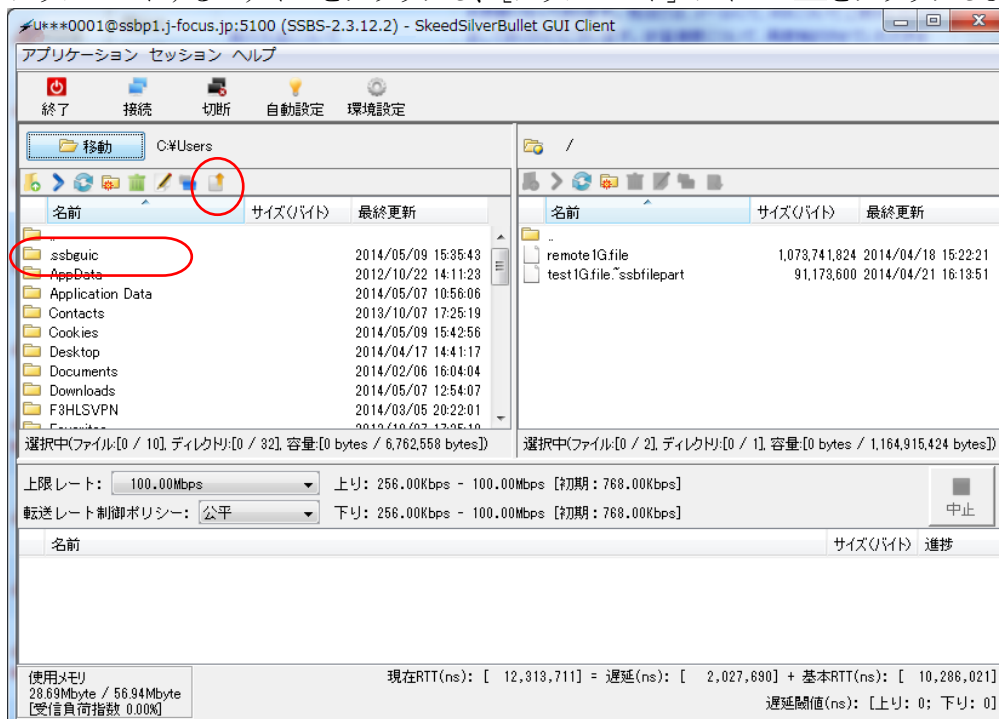


(4) 正常に接続が完了したら、以下のような画面が表示されます。

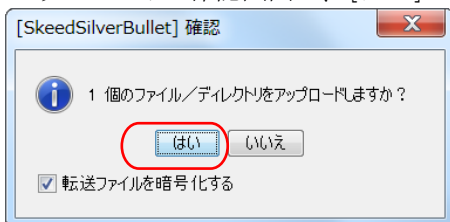


5.2.4. ファイルのアップロード

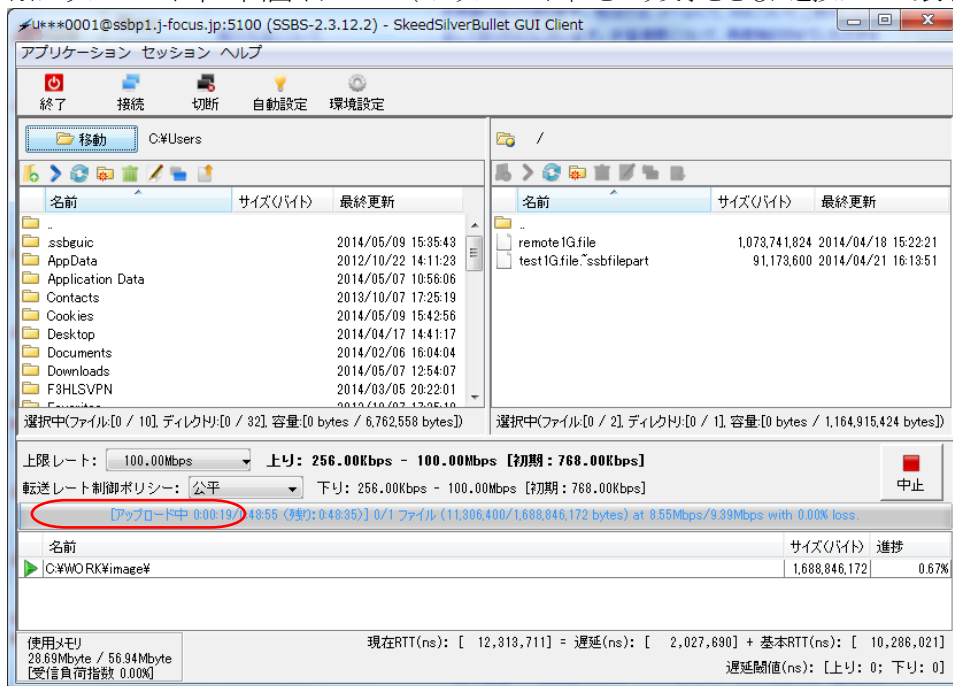
(1) アップロードするファイルをクリックし、[アップロード] アイコンをクリックします。



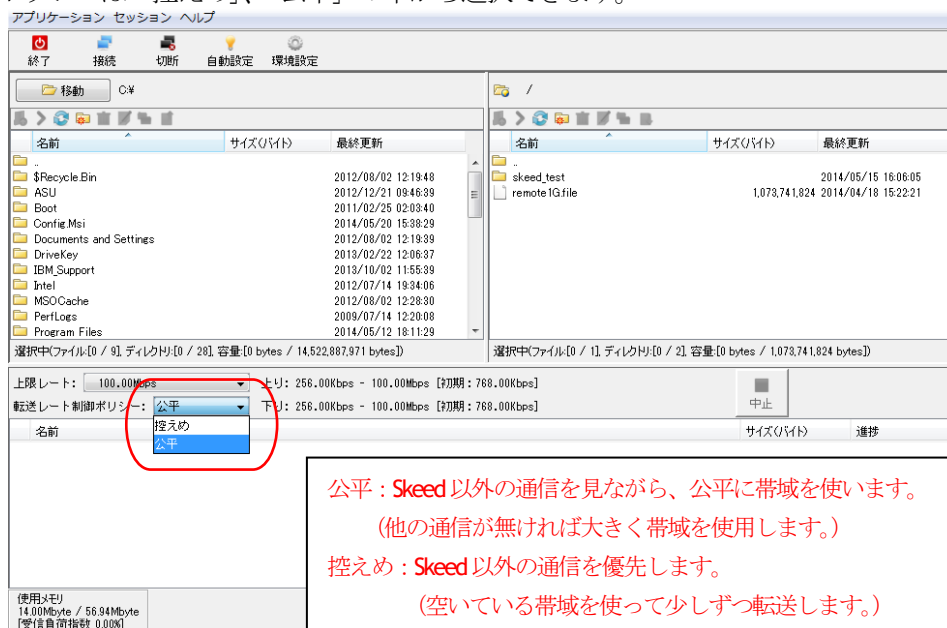
- (2) アップロードの確認画面で、[はい] ボタンをクリックします。



※アップロード中の画面イメージ (“アップロード中”という文字とともに進捗がバーで表示される。)



- (3) 【任意】転送中に「転送レート制御ポリシー」を変更します。
ポリシーは「控えめ」、「公平」の中から選択できます。

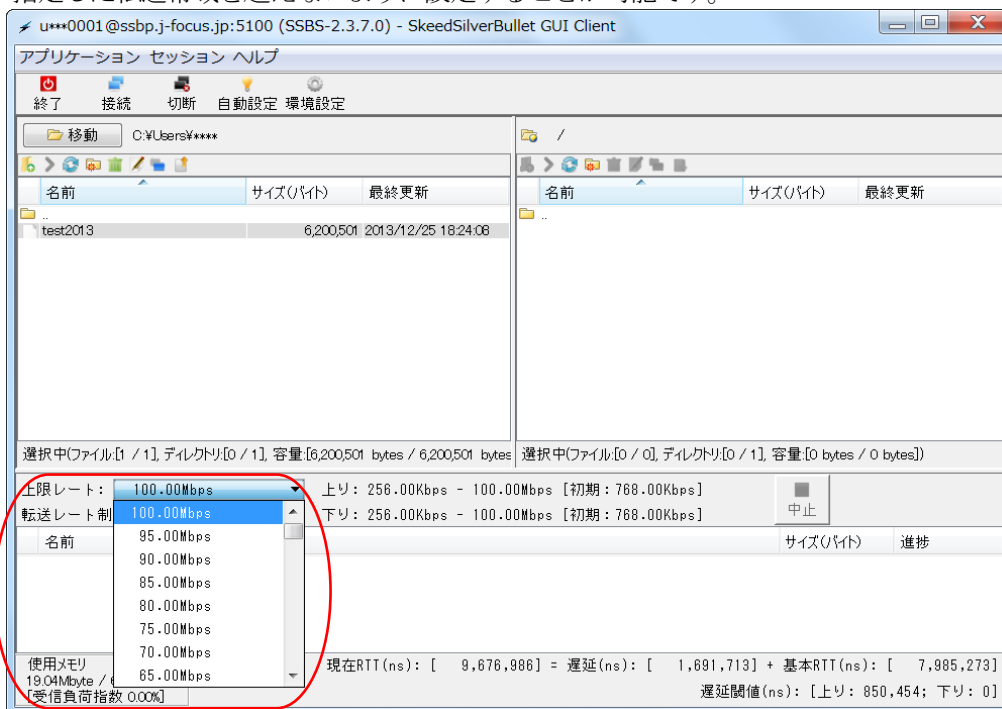


公平: Skeed以外の通信を見ながら、公平に帯域を使います。
(他の通信が無ければ大きく帯域を使用します。)

控えめ: Skeed以外の通信を優先します。
(空いている帯域を使って少しずつ転送します。)

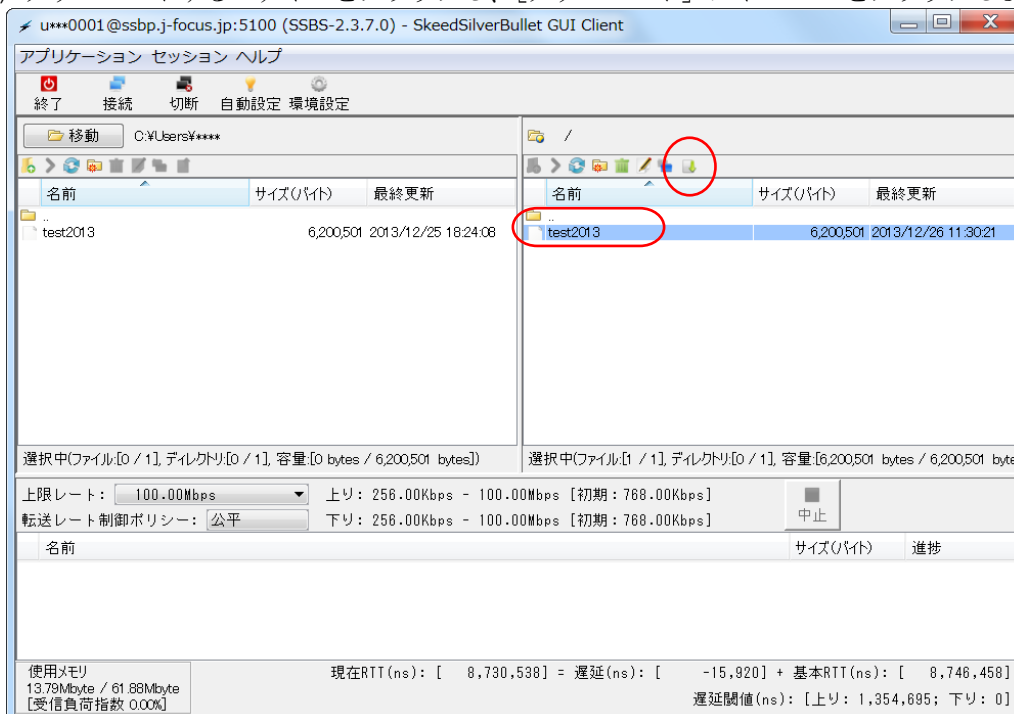
通常は「公平」を選択ください。

- (4) 【任意】転送中に「上限レート」を変更します。
指定した転送帯域を超えないように設定することが可能です。



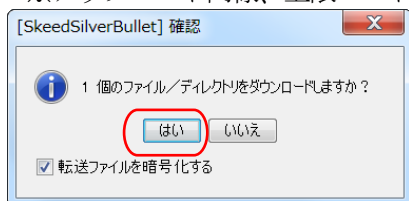
5.2.5. ファイルのダウンロード

- (1) ダウンロードするファイルをクリックし、[ダウンロード] アイコンをクリックします。

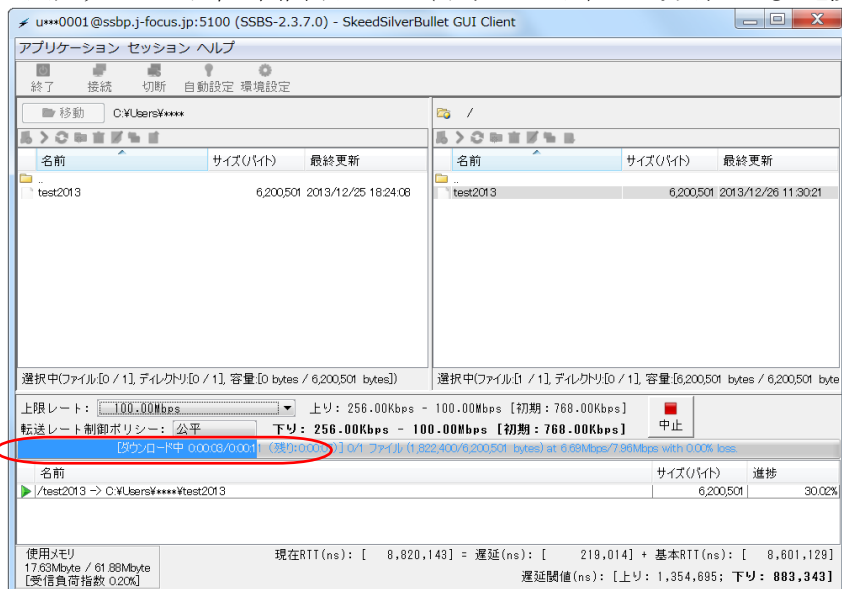


(2) ダウンロード確認画面で、[はい] ボタンをクリックします。

※アップロード同様、上限レート、転送レート制御ポリシーを変更できます。



※ダウンロード中の画面イメージ (ダウンロード中という文字とともに進捗がバーで表示される。)



5.3. Web ブラウザベースの使用手法

5.3.1. システムへのログイン

- (1) Web ブラウザを起動し、以下の URL に接続します。

<https://ssbp1.j-focus.jp:9090/silver-bullet/admin>

または、

<https://ssbp2.j-focus.jp:9090/silver-bullet/admin>

- (2) セキュリティに関する画面で、それぞれ続行するための操作を行います。

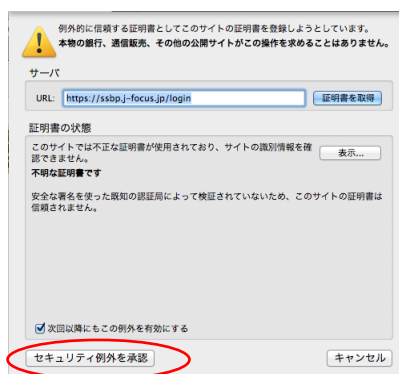
【Internet Explorer の例】[このサイトの閲覧を続行する] ボタンをクリックします。



【Chrome の例】[このまま続行] ボタンをクリックします。

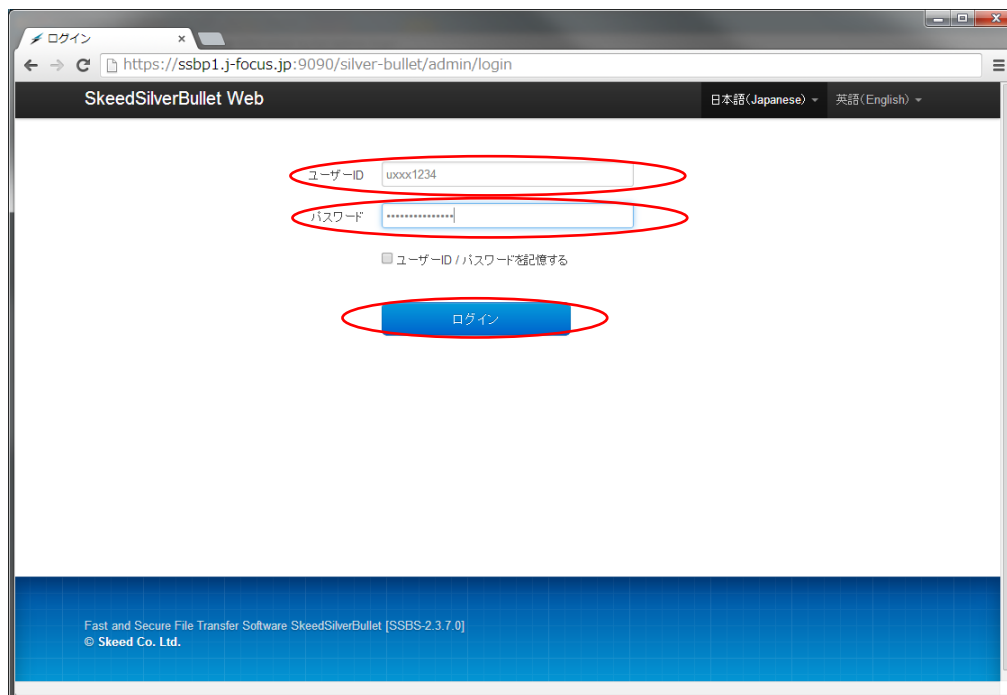


【Firefox の例】[例外を追加] → [セキュリティ例外を承認] を順にクリックします。



(3) 『ログイン』画面にて以下を入力し、[ログイン]ボタンをクリックします。

- ・ユーザーID : アカウント名 (「u」 + “課題名” + 数字 4 桁)
- ・パスワード : アカウントのパスワード



(4) 『ファイル転送>転送操作』画面が表示されることを確認します。

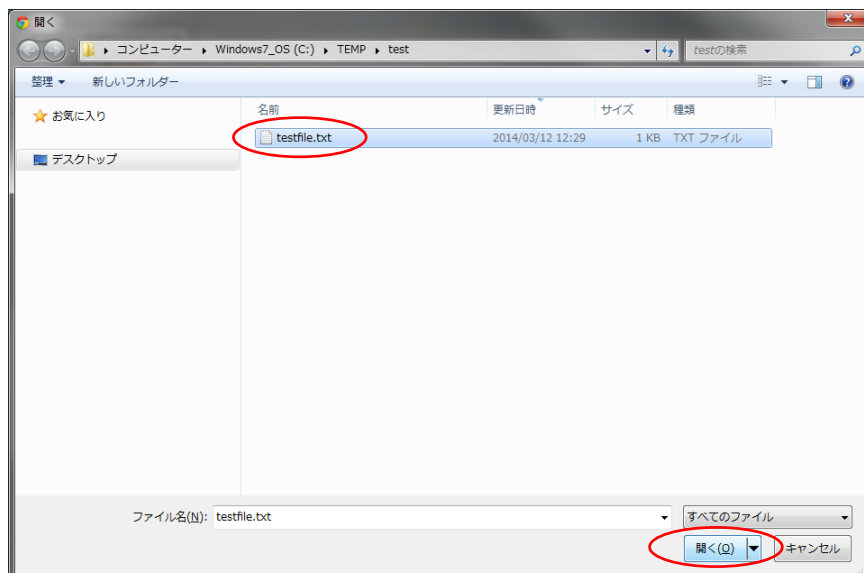


5.3.2. ファイルのアップロード

- (1) 『ファイル転送>転送操作』画面から「アップロード」を選択します。



- (2) 起動したエクスプローラーからアップロードしたいファイルを選択し、「開く」を選択します。
「開く」を選択するとファイル転送が開始されます。



- (3) 『ファイル転送>転送操作』画面に戻り、一覧に転送したファイルが表示されることを確認します。



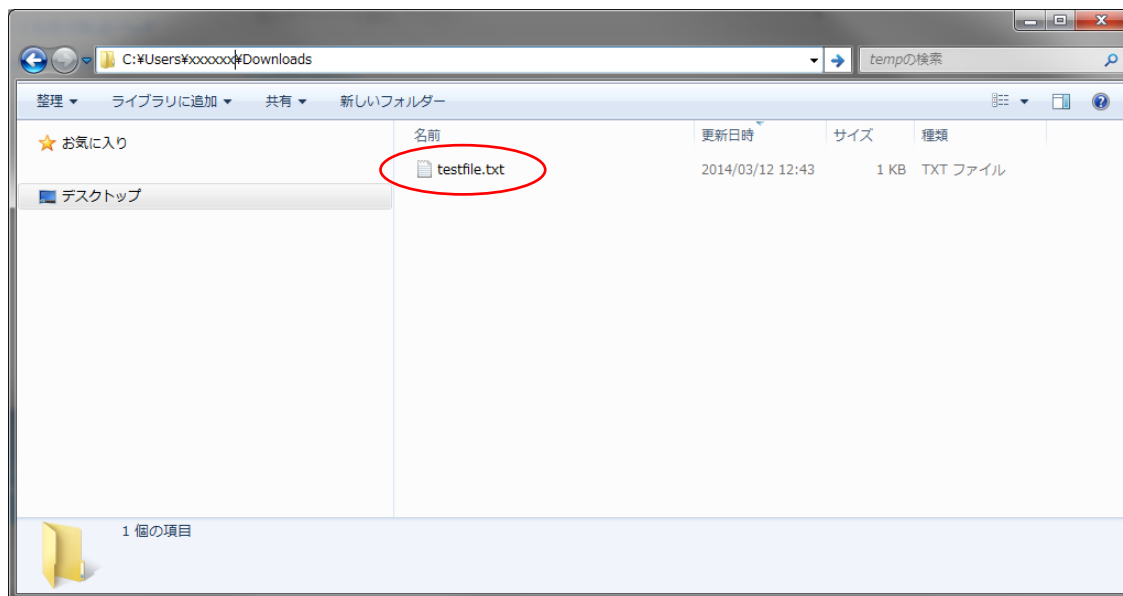
5.3.3. ファイルのダウンロード

- (1) 『ファイル転送>転送操作』画面のホームディレクトリの一覧から、ダウンロードしたいファイル名を選択します。

選択するとブラウザで設定したの保存ディレクトリにファイルがダウンロードされます。



- (2) ブラウザの保存ディレクトリを確認し、ファイルがダウンロードされたことを確認します。



5.3.4. ファイルの削除

- (1) 『ファイル転送>転送操作』画面から削除したいファイル名の左にある「チェックボックス」にチェックを入れ、「削除」を選択します。



- (2) 確認のポップアップ画面がブラウザ内に表示されますので、「OK」を選択し、削除を実行します。

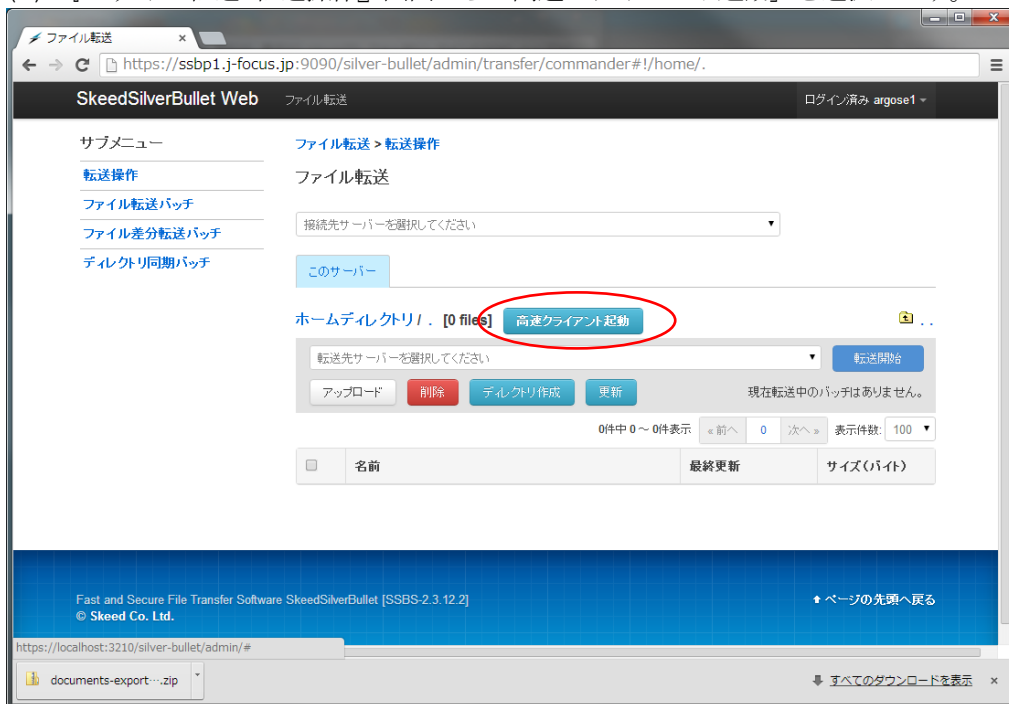


- (3) 『ファイル転送>転送操作』画面に戻り、一覧から削除したファイルが消えていることを確認します。ブラウザ内右上段には実行した内容が通知されます。



5.3.5. Web ブラウザーからの専用クライアント起動

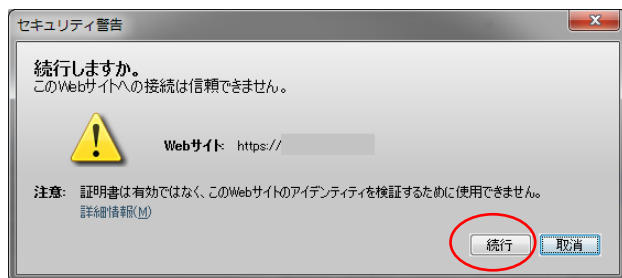
(1) 『ファイル転送>転送操作』画面から「高速クライアント起動」を選択します。



(2) JNLP ファイルのダウンロードを行いますので、保存と実行をしてください。



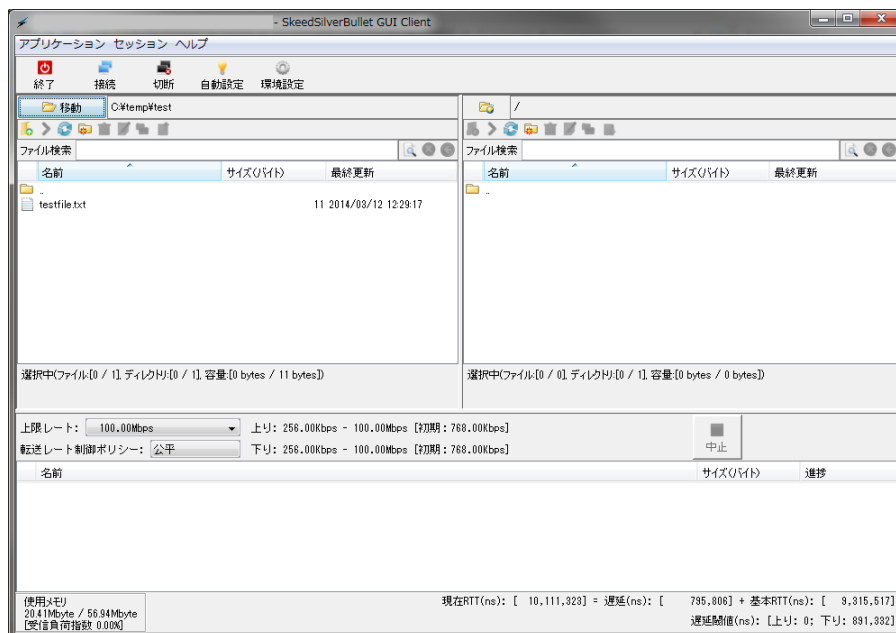
- (3) JAVA からプログラムが実行されます。以下の「セキュリティ警告」がでた場合には「続行」を選択してください。



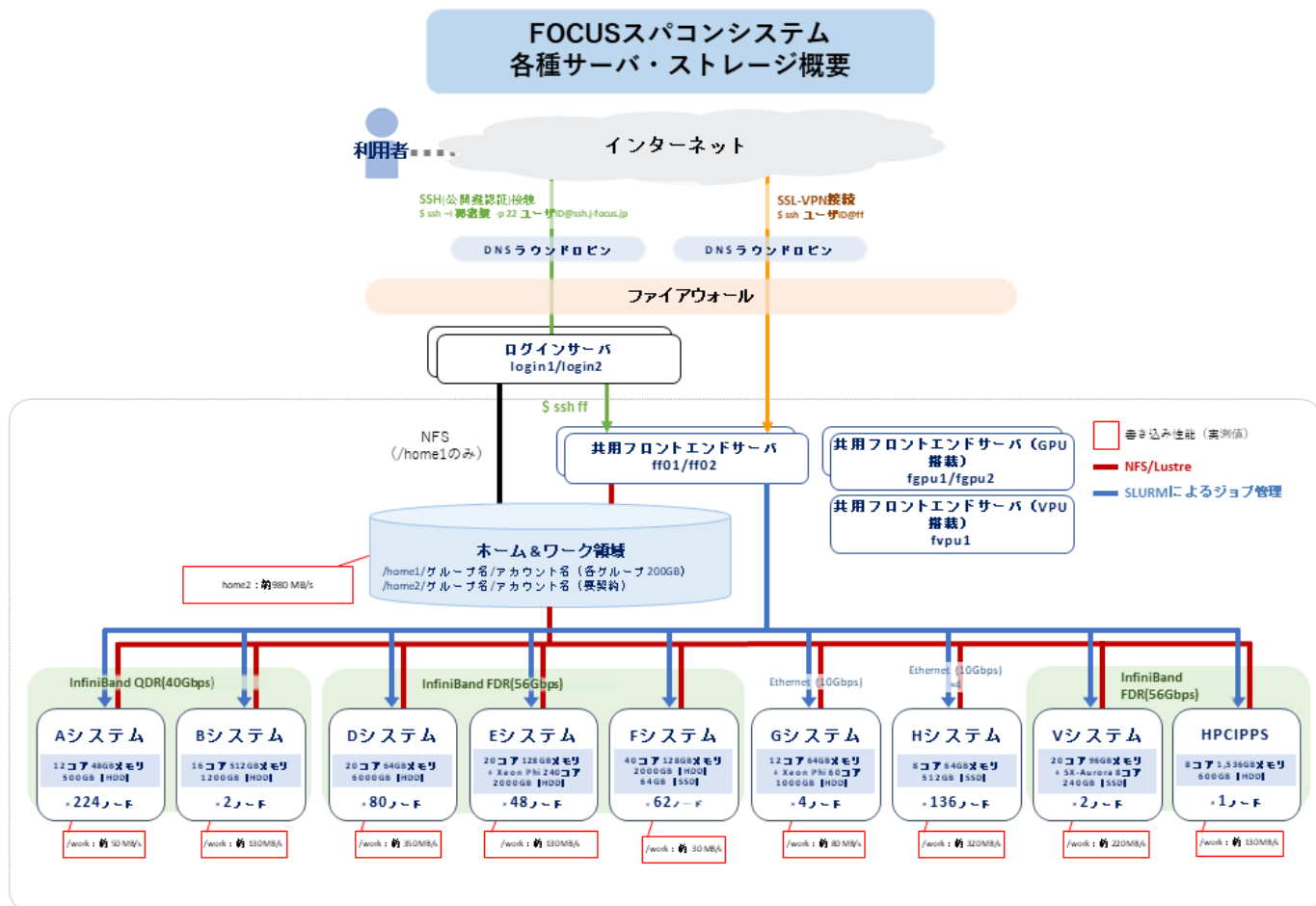
- (4) 「専用クライアント (SkeedSilverBullet GUI)」が起動されます。



Web ブラウザーから起動する場合は、接続先・ユーザ認証が完了した状態となるため、「5.2.3 クライアントの起動」に記載した、下記のログイン後の画面が表示されます。以降の操作は「5.2 専用クライアント (SkeedSilverBullet GUI) の使用方法」を参照してください。



付録 A. FOCUS スパコンシステム各種サーバ・ストレージ概要



各領域の特性

ホーム領域	(/home1)	: NAS ストレージシステム (全体で 500MB/s)
ワーク兼ホーム領域	(/home3)	: 分散ファイルシステム (全体で 11GB/s、プロセスあたり最大 1GB/s)
演算ノードクラッチ領域	(/work)	: ローカルディスク (I/O 性能は各システムにより異なり、他のジョブの影響を受けない。 ジョブ終了時にデータは削除される。)

付録 B. コマンド比較表 (SLURM と LSF)

■ユーザーコマンド	SLURM	LSF
ジョブ実行	sbatch [script_file]	bsub [script_file]
ジョブキャンセル	scancel [job_id]	bkill [job_id]
ジョブ実行状況 (ジョブ毎)	squeue -j [job_id]	bjobs [job_id]
ジョブ実行状況 (ユーザー毎)	squeue -u [user_name]	bjobs -u [user_name]
ジョブのホールド	scontrol hold [job_id]	bstop [job_id]
ジョブのリリース	scontrol release [job_id]	bresume [job_id]
キューリストの表示	squeue	bqueues
ノードリストの表示	sinfo -N または scontrol show node	bhosts
クラスタ全体のステータス	sinfo	bqueues
■環境変数など	SLURM	LSF
ジョブ ID	\$SLURM_JOBID	\$LSB_JOBID
実行ディレクトリ指定	\$SLURM_SUBMIT_DIR	\$LSB_SUBCWD
実行ホストの指定	\$SLURM_SUBMIT_HOST	\$LSB_SUB_HOST
ノードリストの表示	\$SLURM_JOB_NODELIST	\$LSB_HOSTS/LSB_MCPU_HOST
アレイジョブのインデックス表示	\$SLURM_ARRAY_TASK_ID	\$LSB_JOBINDEX
依存ジョブのインデックス表示	\$SLURM_JOB_DEPENDENCY	
■ジョブの詳細	SLURM	LSF
バッチスクリプトのディレクティブ	#SBATCH	#BSUB
キュー (パーティション) 指定	-p [queue]	-q [queue]
実行ノード数	-N [min[-max]]	-n [count]
プロセス数の指定	-n [count]	-n [count]
実行時間の上限指定 (wall time)	-t [min] または -t [days-hh:mm:ss]	-W [hh:mm:ss]
出力ファイル指定	-o [file_name]	-o [file_name]
エラー出力指定	-e [file_name]	-e [file_name]
出力・エラー出力の総合出力	(-e 指定無しで-o を使用)	(-e 指定無しで-o を使用)
イベント通知	--mail-type=[events]	-B または -N
イベント通知先メールアドレス指定	--mail-user=[address[,address]]	-u [address]
ジョブ再投入	--requeue または --no-requeue (未指定時は--no-requeue)	-r
実行ディレクトリ指定	--workdir=[dir_name]	(submission directory)
メモリサイズ指定	--mem=[mem] [M G T] または --mem-per-cpu=[mem] [M G T]	-M [MB]
アカウント名の変更	--account=[account]	-P [account]
ノード当たりのタスク数指定	--tasks-per-node=[count]	—
タスク当たりの CPU 数指定	--cpus-per-task=[count]	—
依存ジョブ	--dependency=[type:job_id]	—
ジョブのプロジェクト化	--wckey=[name]	-P [name]
ジョブ実行ホストの詳細	--odelist=[nodes] または --exclude=[nodes]	-m [nodes]

アレイジョブ	--array=[array_spec]	J "name[array_spec]"
ライセンスの指定	--licenses=[license_spec]	-R "rusage[license_spec]"
開始時間指定	--begin=YYYY-MM-DD[THH:MM[:SS]]	-b [[year:] [month:] daty:] hour:minute

付録c. プログラムとジョブ投入スクリプトのサンプル

FOCUS スパコン利用講習会で使用しているプログラムとジョブ投入スクリプトのサンプルを下記に示します。
※プログラムのコンパイル方法については、「3. コンパイラ、MPI の使用方法」をご参照ください。

1. 逐次ジョブのサンプル

hello_world.c <プログラム>

```
#include <stdio.h>

int main(void)
{
    printf("Hello, world!¥n");
    return 0;
}
```

sample.sh <ジョブ投入スクリプト>

```
#!/bin/bash
#SBATCH -p g006m
#SBATCH -n 1
#SBATCH -J hello_world
#SBATCH -o stdout.%J
#SBATCH -e stderr.%J

./a.out
sleep 60
```

2. スレッド並列ジョブのサンプル

hello_openmp.c <プログラム>

```
#include <stdio.h>
#include <omp.h>

main() {
    #pragma omp parallel
    {
        printf("hello world from %d of %d\n", omp_get_thread_num(), omp_get_num_threads());
    }
}
```

sample_openmp.sh <ジョブ投入スクリプト>

```
#!/bin/bash
#SBATCH -p f006m
#SBATCH -N 1
#SBATCH --ntasks-per-node=1
#SBATCH -c 10
#SBATCH -J hello_openmp
#SBATCH -o hello_openmp_o.%J
#SBATCH -e hello_openmp_e.%J

module load PrgEnv-intel-18.0.3.222

export OMP_NUM_THREADS=${SLURM_CPUS_PER_TASK}
./hello_openmp.out
```

3. プロセス並列ジョブのサンプル

hello_mpi.c <プログラム>

```
#include <stdio.h>
#include "mpi.h"

int main(int argc, char *argv[]) {
    int numprocs, rank, namelen;
    char processor_name[MPI_MAX_PROCESSOR_NAME];
    int iam = 0, np = 1;

    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD, &numprocs);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Get_processor_name(processor_name, &namelen);

    printf("Hello from process %2d out of %2d on %s¥n",
           rank, numprocs, processor_name);

    MPI_Finalize();
}
```

sample_mpi.sh <ジョブ投入スクリプト>

```
#!/bin/bash
#SBATCH -p e006m
#SBATCH -N 2
#SBATCH --ntasks-per-node=12
#SBATCH -c 1
#SBATCH -J hello_mpi
#SBATCH -o hello_mpi_o.%J
#SBATCH -e hello_mpi_e.%J

module load PrgEnv-intel-18.0.3.222
module load MPI-impi-18.3.222

mpirun -np ${SLURM_NTASKS} ./hello_mpi.out
```

4. ハイブリッド並列ジョブのサンプル

hello_hybrid.c <プログラム>

```
#include <stdio.h>
#include "mpi.h"
#include <omp.h>

int main(int argc, char *argv[]) {
    int numprocs, rank, namelen;
    char processor_name[MPI_MAX_PROCESSOR_NAME];
    int iam = 0, np = 1;

    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD, &numprocs);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Get_processor_name(processor_name, &namelen);

#pragma omp parallel default(shared) private(iam, np)
    {
        np = omp_get_num_threads();
        iam = omp_get_thread_num();
        printf("Hello from process %2d out of %2d from thread %2d out of %3d on %s¥n",
              rank, numprocs, iam, np, processor_name);
    }

    MPI_Finalize();
}
```

hello_hybrid.sh <ジョブ投入スクリプト>

```
#!/bin/bash
#SBATCH -p e006m
#SBATCH -N 2
#SBATCH --ntasks-per-node=4
#SBATCH -c 5
#SBATCH -J hello_hybrid
#SBATCH -o hello_hybrid_o.%J
#SBATCH -e hello_hybrid_e.%J

module load PrgEnv-intel-18.0.3.222
module load MPI-impi-18.3.222

export OMP_NUM_THREADS=${SLURM_CPUS_PER_TASK}
mpirun -np ${SLURM_NTASKS} ./hello_hybrid.exe
```