［大規模科学計算システム］

# サブシステム AOBA-S の利用法

情報部デジタルサービス支援課 共同利用支援係 共同研究支援係

## 1　はじめに

本センターはスーパーコンピュータ AOBA のサブシステム AOBA-S の運用を 2023 年 8 月から開始しています。本稿では、サブシステム AOBA-S のプログラミング利用ガイドとして、AOBA-S 用フロントエンドサーバへのログイン、プログラムの作成からコンパイル、およびジョブ実行等の使い方についてご紹介します。

サイバーサイエンスセンターの大規模科学計算システムを利用するためには利用申請が必要です。利用申請について詳しくは以下をご参照ください。

【利用申請】https://www.ss.cc.tohoku.ac.jp/apply-for-use/

## 2　システムの構成

本センターでは、スーパーコンピュータ AOBA として、サブシステム AOBA-S（SX-Aurora TSUBASA Type 30A）と、サブシステム AOBA-A（SX-Aurora TSUBASA Type 20B）、およびサブシステム AOBA-B（LX 406Rz-2）の 3 つの計算機システムをサービスしています（図 1）。また、AOBA-S 用のストレージとして 4.5PB、AOBA-A および AOBA-B 用のストレージとして 2PB のストレージシステムをサービスしています。利用者は SINET6 を利用したリモートアクセス接続により、全国から大規模科学計算システムを利用することが可能です。

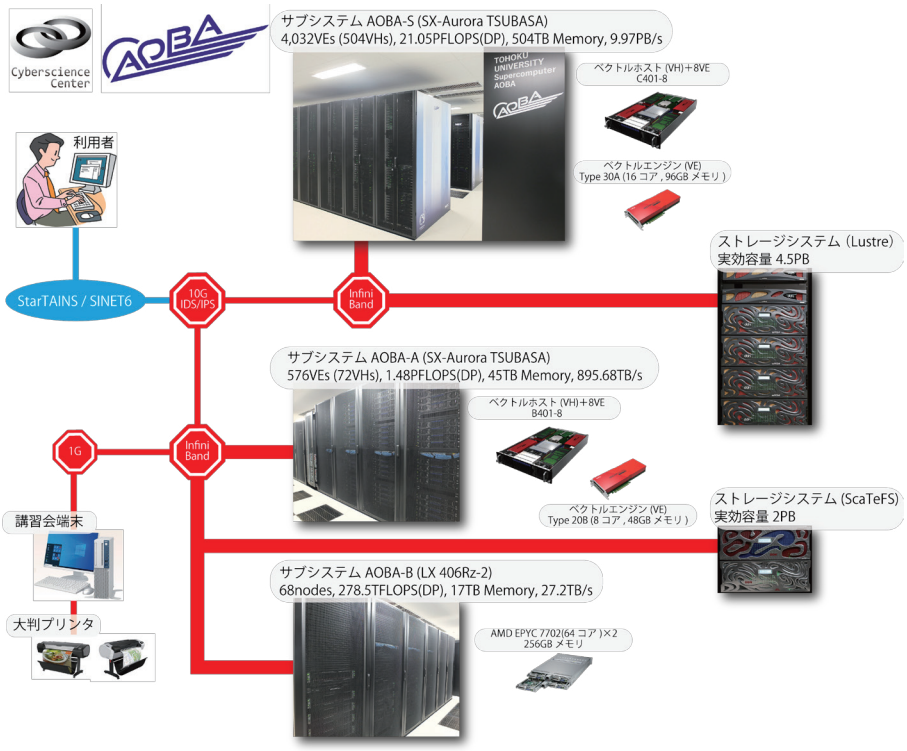


図 1　スーパーコンピュータ AOBA の構成

## 2.1 サブシステム AOBA-S の特徴

### 2.1.1 SX-Aurora TSUBASA アーキテクチャ

サブシステム AOBA-S はサブシステム AOBA-A と同じくベクトルアーキテクチャを継承しています。アプリケーション演算処理を行うベクトルエンジン（以下、VE）部と、主に OS 処理を行うベクトルホスト（以下、VH）部により構成されます。PCIe カードに搭載される VE 部はベクトルプロセッサ、および高速メモリから構成され、x86/Linux である VH と PCIe 経由で接続されます。

1VE は理論最大演算性能 4.91TFLOPS(DP) となる 16 コアベクトルプロセッサを 1 基、主記憶は 96GB を搭載し、2.45TB/s という高いメモリバンド幅でプロセッサと接続されることで、高い演算性能とメモリ性能の最適化を実現しています。

今回導入した AOBA-S システムは、1VH と 8VE が構成単位となる C401-8 モデルを採用し、システム全体では 504 個の VH と 4,032 個の VE で構成されます。VE と VH を合わせたシステム全体の理論演算性能は、21.05PFLOPS(DP) 、主記憶は 504TB、総メモリバンド幅は 9.97PB/s となります。


図 2　サブシステム AOBA-S（32 ラック）

- マルチコアベクトルエンジン（16 コア）あたり、4.91TFLOPS(DP) の理論演算性能
- 1VE あたり 96GB の共有メモリを搭載し、メモリバンド幅（データ転送性能）は 2.45TB/s

### 2.1.2 ベクトルプロセッサ

ベクトルプロセッサとは、ベクトル演算を行う専用のハードウェアを持つコンピュータです。ベクトル演算は、ループ中で繰り返し処理されるような配列データの演算に対して一括して演算を実行するため、高速に演算することができます。

ベクトル計算機は科学技術用の数値計算に適しており、大量のデータを繰り返し処理するような大規模計算に向いていると言われています。本センターでは、流体解析や気象解析、電磁界解析をはじめとする大規模シミュレーションに利用されます。

#### 2.1.3　ソフトウェア

Linux OS 環境上で、ベクトルエンジンの開発環境を利用できます。ベクトル処理、並列処理に対応した大規模プログラムの作成と実行が可能です。

**■プログラミング言語**　GNU 互換環境を装備し、アプリケーションの実効性能を向上させる高度な自動ベクトル化・自動並列化機能を備えた Fortran/C/C++ コンパイラが利用できます。それぞれのコンパイラは自動ベクトル化機能、自動並列化機能を有していますので、既存のプログラムを修正することなくベクトル化、並列化することができます。自動並列化機能と OpenMP による共有メモリ並列実行と、システム構成に最適化された MPI ライブラリにより、分散メモリ並列実行も可能です。

**■科学技術計算ライブラリ**　NEC Numeric Library Collection（NLC）は業界標準の BLAS、FFTW、LAPACK、ScaLAPACK を含む、最適化された科学技術計算ライブラリです。NLC は広範な分野の数値シミュレーションプログラムの作成を強力に支援する数学ライブラリのコレクションで、VE での実行に最適化されています。NLC を用いることにより、難解な数値計算アルゴリズムの詳細に煩わされることなく高度な科学技術計算プログラムを作成することができ、数値シミュレーションプログラム開発の生産性を大幅に改善することができます。NLC は、Fortran または C 言語プログラムから利用できます。

### 2.2　プログラムの実行方法

プログラムの実行の方法として表 1 の 4 種類が利用できます。並列実行には 3 つの手法があります。

**■逐次実行**　単一のコアで動作するプログラムです。VE 内には 16 コアがありますが、逐次実行では 1 コアのみが利用されます。メモリは 96GB まで利用できます。

**■自動並列実行**　逐次処理プログラムを、コンパイラが並列化可能な箇所を自動的に判断して並列化します。プログラムを改めて書き直す必要はなく、既存のプログラムをそのまま利用することができます。VE 内の 16 コアまでの並列実行が可能で、メモリは 96GB まで利用できます。

**■OpenMP 並列実行**　自動並列化と同じく逐次処理プログラムを並列化します。並列化の判断は自動ではなくユーザが明示的に行います。ソース中の並列化したい箇所に並列化指示行を追加します。VE 内の 16 コアまでの並列実行が可能で、メモリは 96GB まで利用できます。

**■MPI 並列実行**　MPI ライブラリによりプロセッサ間通信を行います。データの分割、処理方法等の並列処理手順を明示的に記述した MPI プログラムを作成する必要があります。分散メモリ並列実行が可能なので、複数の VE を用いた実行が可能です。

センターでは通常のサービスとして最大 2,048VE（32,768 コア）、メモリは 192TB まで利用できます。MPI 並列と自動並列/OpenMP 並列を同時に利用した並列実行も可能です。

表 1　実行の種類、特徴、最大並列数、最大メモリ量

| 実行の種類 | 並列化の方法 | コードの改変量 | 最大並列数 | 最大メモリ量 |
|---|---|---|---|---|
| 逐次実行 | - | - | 1 コア | 96GB |
| 自動並列化 | コンパイラによる自動並列化 | なし | 16 コア | 96GB |
| OpenMP 並列 | ユーザによる指示行挿入 | 少ない | 16 コア | 96GB |
| MPI 並列 | MPI ライブラリを用いたプログラミング | 多い | 32,768 コア | 192TB |

# 3　利用者向けサーバへのログイン方法

AOBA-S 用のフロントエンドサーバ、およびデータ転送サーバへのログインは、AOBA-A,B 用のログインサーバと同じく、公開鍵認証方式による SSH 接続を採用しています。

公開鍵認証方式で使用する鍵ペアの作成方法と、各サーバへのログイン方法についてご紹介します。解説では SSH 接続に以下のターミナル（端末）ソフトを使用する例をご紹介します。

- （Windows の場合） Windows PowerShell
- （macOS ／ Linux の場合） ターミナル

本センターのシステムをはじめて利用する方は以下の手続きが必要です。

(1) 利用者番号の取得（利用申請：https://www.ss.cc.tohoku.ac.jp/apply-for-use/ ）
(2) 鍵ペアの作成（3.3 節）

AOBA-A および AOBA-B を利用していた方は、(1) (2) の手続きは不要です。以前使用していた利用者番号および鍵ペアをそのままご利用いただけます。3.4 節からお読みください。

## 3.1　利用者向けサーバ名と用途

表 2 に AOBA-S 用の利用者向けサーバ名とホスト名、および用途を示します。

表 2　AOBA-S 用の利用者向けサーバ

| サーバ名 | ホスト名 | 用途 |
|---|---|---|
| フロントエンドサーバ | sfront.cc.tohoku.ac.jp | コンパイル作業、AOBA-S へのジョブ投入<br>ローカル PC との小規模なデータ転送 |
| データ転送サーバ | sfile.cc.tohoku.ac.jp | ローカル PC との大規模なデータ転送<br>AOBA-A,B 用ストレージとのデータ転送 |
| HPCI 用フロントエンドサーバ | shpcif.cc.tohoku.ac.jp | HPCI、HPCI-JHPCN 課題用フロントエンドサーバ |

### 3.2　SSH 認証鍵ペアの作成とログインまでの概要

図 3 に、鍵ペアの作成からログインまでの流れを示します。

1. 鍵ペアの作成は 3.3 節で
2. ターミナルソフトの設定および 3. ログイン については 3.4 節で

それぞれ解説します。

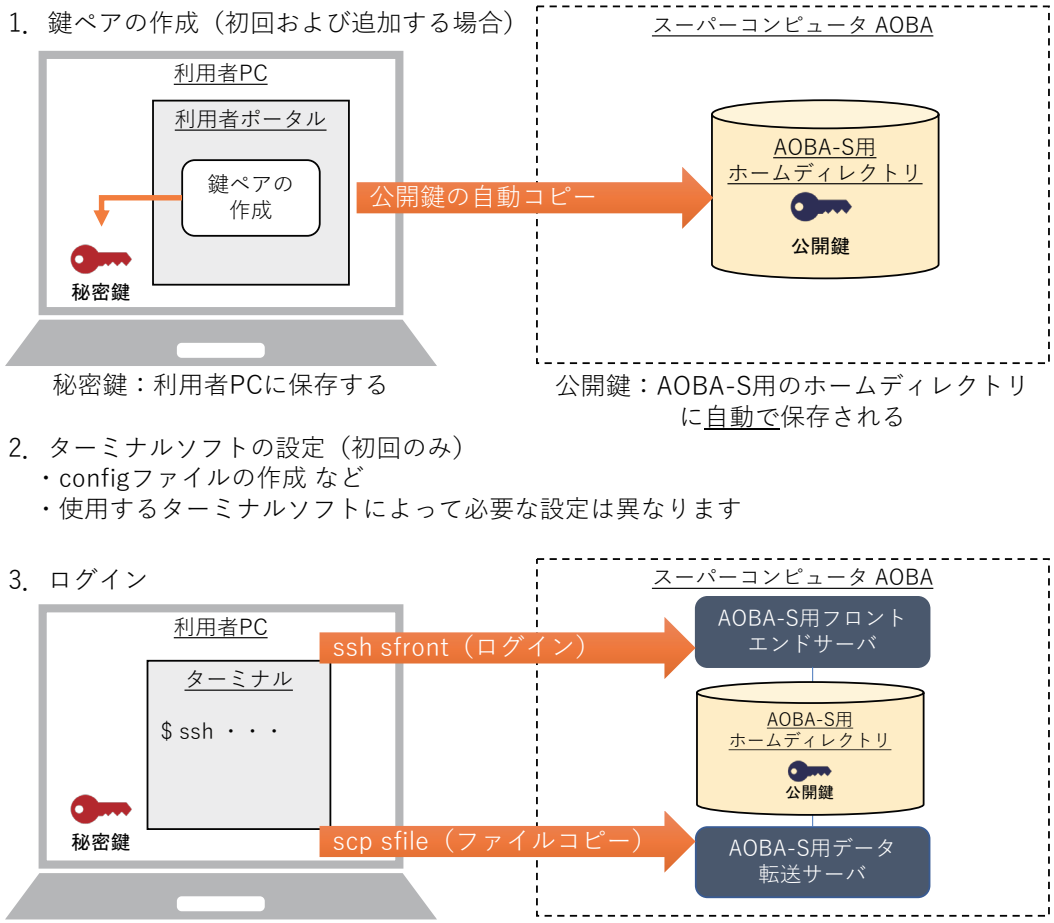


図 3　鍵ペアの作成からログインまで

1. 鍵ペアの作成（初回ログイン時、およびログイン端末を追加する場合）

利用者ポータルで鍵ペアを作成します。作成された秘密鍵は利用者のローカル PC に保存します。公開鍵は、スーパーコンピュータ AOBA のホームディレクトリ上に自動で保存されます。

【利用者ポータル】https://www.ss.cc.tohoku.ac.jp/portal/

2. ターミナルソフトの設定（初回ログイン時）

各利用者用サーバにログインするための設定を行います。使用するターミナルソフトによって必要な設定が異なります。

3. ログイン

利用者のローカル PC に保存した秘密鍵を使って利用者用サーバにログインします。

## 3.3 公開鍵認証方式で使用する鍵ペアの作成

### 3.3.1 公開鍵認証方式を使用する上での注意事項

**以下の注意事項を必ず守って**ください。

守られない場合、不正アクセス（不正ログイン、クライアントのなりすまし、暗号化された通信の暴露、他サーバへの攻撃など）のリスクが非常に高まるので**大変危険**です。

- パスフレーズなしの秘密鍵を使用しないこと
- 秘密鍵、パスフレーズを使いまわさないこと
- 秘密鍵を持ち出さないこと（メールに添付しない、USB メモリ等に保存しない）
- 秘密鍵をスーパーコンピュータ AOBA のホームディレクトリに保存しないこと
- 公開鍵と秘密鍵の鍵ペアを同一ホスト上に保存しないこと

### 3.3.2 鍵ペアの作成（初回ログイン時、および接続する PC を追加する場合）

**■初回ログイン時**　鍵ペアの作成は利用者ポータルで行います。

(1) 以下の URL から利用者ポータルに接続します。利用者ポータルには利用者番号と LDAP パスワード（※）でログインします。
【利用者ポータル】https://www.ss.cc.tohoku.ac.jp/portal/
(2)「SSH 公開鍵登録」ボタンをクリックします。
(3) 利用者ポータルの画面の説明に従い、鍵ペアを作成します。
（パスフレーズを設定し、鍵生成・登録ボタンをクリック）
(4) 作成された秘密鍵を利用者のローカル PC に保存します。保存先は以下を推奨します。フォルダがない場合は新規作成します。
  - （Windows の場合） C:¥Users¥（ユーザ名）¥.ssh
  - （macOS ／ Linux の場合） $HOME/.ssh

ポータルサイトで作成された公開鍵は、AOBA-S と AOBA-A,B のホームディレクトリの公開鍵ファイル（$HOME/.ssh/authorized_keys）に自動で保存されます。

※利用者ポータルで使用する LDAP パスワードの変更方法は、3.6 節を参照してください。

**■別の PC からログインする場合**　接続する PC からポータルサイトにアクセスし、新しい鍵ペアを作成します。既存の秘密鍵を使いまわすのではなく、端末ごとに鍵ペアを作成してください。

## 3.4 利用者向けサーバへのログイン方法

### 3.4.1 ターミナルソフトの設定（各 PC での設定）

利用者の PC 上で、ターミナルソフトの設定を行います。以降の解説では、次のフォルダを「.ssh フォルダ」と呼び、秘密鍵を「id_rsa_cc」というファイル名で.ssh フォルダに保存した場合とします。

- （Windows の場合） C:¥Users¥（ユーザ名）¥.ssh
- （macOS ／ Linux の場合） $HOME/.ssh

各利用者向けサーバのホスト名は、次の文字列で設定するものとして解説します。ホスト名には任意の文字列を設定することができますが、他の文字列を設定した場合は、以降の解説におけるホスト名を読み替えてください。

- （AOBA-S 用フロントエンドサーバ） sfront
- （AOBA-S 用データ転送サーバ） sfile
- （AOBA-S HPCI 用フロントエンドサーバ） shpcif

(1) macOS ／ Linux の場合は、秘密鍵のパーミッションを 600 に変更が必要です。ターミナルソフトを起動し以下のコマンドを実行します（リスト 1)。

リスト 1　秘密鍵のパーミッション変更

```
(localhost)$ chmod 600 ${HOME}/.ssh/id_rsa_cc
```

以降は Windows、macOS ／ Linux 共通です。

(2) .ssh フォルダの「config」というファイルをテキストエディタで開きます。ファイルがない場合は新規作成します。拡張子は付けません。（フォルダの設定を「拡張子を表示しない」にしている場合、意識せずに拡張子つきのファイルを作成している可能性があります。config ファイルに拡張子がついていると正しく設定が読み込まれません。）

(3) config ファイルにリスト 2 に示した設定を記述します。ホスト名を例とは別に設定した場合は、以降のコマンドではご自身の環境に合わせて読み替えてください。

リスト 2　config ファイルの設定方法

```
# AOBA-S用フロントエンドサーバに接続するための設定
Host sfront                          # AOBA-S用フロントエンドサーバ名をsfrontで指定
HostName sfront.cc.tohoku.ac.jp      # ホスト名をFQDNで指定
User 利用者番号                      # 利用者番号を指定
IdentityFile $HOME/.ssh/id_rsa_cc    # 秘密鍵の保存場所とファイル名を指定

# AOBA-S用データ転送サーバに接続するための設定
Host sfile                           # AOBA-S用データ転送サーバ名をsfileで指定
HostName sfile.cc.tohoku.ac.jp       # ホスト名をFQDNで指定
User 利用者番号                      # 利用者番号を指定
IdentityFile $HOME/.ssh/id_rsa_cc    # 秘密鍵の保存場所とファイル名を指定

# HPCI用フロントエンドサーバに接続するための設定
Host shpcif                          # HPCI用フロントエンドサーバ名をshpcifで指定
HostName shpcif.cc.tohoku.ac.jp      # ホスト名をFQDNで指定
User 利用者番号                      # 利用者番号を指定
IdentityFile $HOME/.ssh/id_rsa_cc    # 秘密鍵の保存場所とファイル名を指定
```

### 3.4.2　AOBA-S 用フロントエンドサーバへのログイン

ターミナルソフトを起動しリスト 3 に示したコマンドを実行すると config ファイルに記述した設定が読み込まれ、AOBA-S 用のフロントエンドサーバにログインします。

リスト 3　AOBA-S 用フロントエンドサーバへのログイン

```
(localhost)$ ssh sfront
Last login: Tue Aug 1 12:34:56 2023 from x.x.x.x
(sfront)$ # 接続するとプロンプトのホスト名が変わります
```

フロントエンドサーバは冗長構成になっており、自動的に sfront1 または sfront2 が選択されます。どちらにログインしても環境は変わりません。

なお、フロントエンドサーバでは一定時間以上のプロセスは実行できません。また、大容量のデータ転送はシステムに高い負荷がかかります。大容量のデータ転送を行う場合はデータ転送サーバをご利用ください。

### 3.4.3 AOBA-S 用データ転送サーバの利用方法

AOBA-S 用のデータ転送サーバは、利用者のローカル PC と AOBA-S 用のストレージ間のデータ転送、および AOBA-S 用ストレージと AOBA-A,B 用ストレージ間のデータコピーに用います。

ローカル PC とのデータ転送は、scp コマンドや sftp コマンド、ファイル転送のアプリケーションを用いて行います。リスト 4 に scp コマンドを使用したデータ転送の例を示します。ターミナルソフトを起動し、scp コマンドでローカル PC と AOBA-S 用ストレージ間のデータ転送を行います。

リスト 4　AOBA-S 用データ転送サーバ

```
# ローカルPCのデータをAOBA-S用ストレージに転送する場合
(localhost)$ scp -r ローカルPCのデータ sfile:/uhome/利用者番号/コピー先

# AOBA-S用ストレージのデータをローカルPCに転送する場合
(localhost)$ scp -r sfile:/uhome/利用者番号/データ ローカルPCのコピー先
```

また AOBA-S 用ストレージと AOBA-A,B 用ストレージ間のデータコピーは、AOBA-S 用のデータ転送サーバにログインした後に cp コマンドで行います。リスト 5 に AOBA-S 用のデータ転送サーバへのログイン方法、およびリスト 6 に cp コマンドを使用したデータコピーの例を示します。

ターミナルソフトを起動しリスト 5 示したコマンドを実行すると config ファイルに記述した設定が読み込まれ、AOBA-S 用のデータ転送サーバにログインします。

リスト 5　AOBA-S 用データ転送サーバ

```
(localhost)$ ssh sfile
Last login: Tue Aug 1 12:34:56 2023 from x.x.x.x
(sfile)$ # 接続するとプロンプトのホスト名が変わります
```

AOBA-S 用データ転送サーバ上でのマウントポイントは以下の通りです。

- （AOBA-S 用ストレージ）　/uhome/利用者番号/
- （AOBA-A,B 用ストレージ）/mnt/stfs/uhome/利用者番号/

sfile にログインした後リスト 6 に示したコマンドで、AOBA-S 用ストレージと AOBA-A,B 用ストレージ間のデータコピーを行います。

リスト 6　AOBA-S 用データ転送サーバ

```
# AOBA-A,B用ストレージのデータをAOBA-S用ストレージにコピーする場合
(sfile)$ cp /mnt/stfs/uhome/利用者番号/AOBA-A,Bのデータ /uhome/利用者番号/AOBA-Sのコピー先

# AOBA-S用ストレージのデータをAOBA-A,Bストレージにコピーする場合
(sfile)$ cp /uhome/利用者番号/AOBA-Sのデータ /mnt/stfs/uhome/利用者番号/AOBA-A,Bのコピー先
```

なお、AOBA-S 用フロントエンドサーバと HPCI 用フロントエンドサーバからは、AOBA-A,B 用ストレージ用のマウントポイントにはアクセスできません。

### 3.4.4 HPCI 用フロントエンドサーバへのログイン

ターミナルソフトを起動しリスト 7 に示したコマンドを実行すると config ファイルに記述した設定が読み込まれ、HPCI 用のフロントエンドサーバにログインします。HPCI および HPCI-JHPCN 課題利用者のみログイン可能です。

リスト 7　HPCI 用フロントエンドサーバ

```
(localhost)$ ssh shpcif
Last login: Tue Aug 1 12:34:56 2023 from x.x.x.x
(shpcif)$ # 接続するとプロンプトのホスト名が変わります
```

## 3.5　ログインシェルの確認と変更

ログインシェルのデフォルトは bash が設定されています。設定の確認および変更はリスト 8 に示した手順で行います。ログインシェルの変更がシステム全体に反映されるまで、15 分程度かかります。

ログインシェルの変更は、AOBA システム（AOBA-S 用、AOBA-A,B 用の全てのホスト、プリンタサーバ）の利用者向けサーバに対して設定されます。

リスト 8　ログインシェルの確認と変更

```
(sfront)$ fchsh                    （現在のログインシェルの確認）
Enter Password:                    （LDAPパスワードを入力）
loginShell: /bin/bash              （現在のログインシェルが表示される）

(sfront)$ fchsh /bin/tcsh          （ログインシェルを/bin/tcshに変更）
Enter Password:                    （LDAPパスワードを入力）
Changed loginShell to /bin/tcsh    （ログインシェルが変更された）
```

## 3.6　LDAP パスワードの変更

利用者ポータルやログインシェルの変更などで使用する LDAP パスワードの変更は、利用者ポータルで行います。

(1) 以下の URL から利用者ポータルにログインします。利用者ポータルには、利用者番号と現在の LDAP パスワードでログインします。
【利用者ポータル】https://www.ss.cc.tohoku.ac.jp/portal/
(2)「パスワード変更」ボタンをクリックします。
(3) 利用者ポータルの画面の説明に従い、新しい LDAP パスワードを設定します。
(4) 以下で使用するパスワードが変更されます。
- 利用者ポータルへのログイン
- 大判カラープリンタのプリンタサーバへのログイン
- ログインシェルの変更時

# 4　プログラムのコンパイルと実行

本章ではプログラムのコンパイルと、サブシステム AOBA-S で実行する手順を説明します。

## 4.1　AOBA-S 用フロントエンドサーバへのログイン

サブシステム AOBA-S 用プログラムのコンパイルとジョブ投入は AOBA-S 用フロントエンドサーバで行います。AOBA-S 用フロントエンドサーバへの詳しい接続方法は 3.4 節をご参照ください。設定が完了している場合はリスト 9 に示したコマンドで、AOBA-S 用フロントエンドサーバにログインします。

なお AOBA-S 用フロントエンドサーバから、AOBA-A および AOBA-B へのジョブ投入などの操作はできません。

リスト 9　AOBA-S 用フロントエンドサーバへのログイン

```
(localhost)$ ssh sfront
Last login: Tue Aug 1 12:34:56 2023 from x.x.x.x
(sfront)$ # 接続するとプロンプトのホスト名が変わります
```

## 4.2　ソースコードの作成

ソースコードを作成する代表的な方法は以下の通りです。

- フロントエンドサーバ上でテキストエディタで作成する
- ローカル PC 上のテキストエディタで作成し、AOBA-S 用ストレージにファイル転送する
- Visual Studio Code などの開発環境で作成する

フロントエンドサーバのコンソール上でソースファイルを作成するためのテキストエディタは、emacs や vim、nano が利用できます。

ローカル PC で作成したソースコードファイルを転送する場合は、ファイル転送ソフトや scp コマンドで利用者のホームディレクトリに転送してください。その際、ソース（テキスト）ファイルは ASCII モードで転送します。

Windows 環境で作成したソースコードの改行コードと文字コードを Linux 用に変換するためには、AOBA-S 用ストレージに転送したソースコードに nkf コマンドを利用します（リスト 10）。

リスト 10　nkf コマンドの使用例

```
# 改行コードをLF、文字コードをUTF-8に変換し、別のファイルに書き出す方法
(sfront)$ nkf -Lu ソースコードファイル名 > 変換後ファイル名

# 改行コードをLF、文字コードをUTF-8に変換して上書き保存する方法
(sfront)$ nkf --overwrite -Lu ソースコードファイル名
```

開発環境の利用方法については各アプリケーションのマニュアルをご参照ください。

## 4.3　コンパイル

### 4.3.1　コンパイラの仕様

AOBA-S では SX-Aurora TSUBASA 用に最適化された Fortran コンパイラ、および C/C++ コンパイラを利用できます。コンパイラの仕様は以下の通りです。

Fortran コンパイラ

- nfort、mpinfort コマンドでコンパイルとリンク
- 自動ベクトル化・自動並列化機能を実装
- ISO/IEC 1539-1:2010 Programming languages - Fortran に準拠
- OpenMP Application Program Interface Version 4.5 に準拠
- ISO/IEC 1539-1:2018 Programming languages - Fortran の一部機能にも対応
- OpenMP Application Program Interface Version 5.0 の一部機能にも対応

C/C++ コンパイラ

- ncc、mpincc、nc++、mpinc++ コマンドでコンパイルとリンク
- 自動ベクトル化・自動並列化機能を実装
- ISO/IEC 9899:2011 Programming languages - C に準拠
- ISO/IEC 14882:2014 Programming languages - C++ に準拠
- ISO/IEC 14882:2017 Programming languages - C++ に準拠
- OpenMP Application Program Interface Version 4.5 に準拠
- ISO/IEC 14882:2020 Programming languages - C++ の一部機能にも対応
- OpenMP Application Program Interface Version 5.0 の一部機能にも対応

### 4.3.2　コンパイルの手順

ソースコードファイルはそれぞれの言語用コマンドでコンパイルを行います。以下では単一コアで実行する逐次実行、自動並列化または OpenMP 並列化による共有メモリ並列実行、および MPI ライブラリによる分散メモリ並列実行のコンパイル手順について解説します。

**■Fortran プログラムのコンパイル**　ソースコードファイルは nfort コマンドまたは mpinfort コマンドでコンパイルを行います。必要に応じて、表 3 に示したコンパイルオプションをコンパイル時に指定します。その他のコンパイルオプションについては、6 章のコンパイラユーザーズガイドをご参照ください。

ソースファイルの拡張子は、自由形式（フリーフォーマット）なら.f90 .f95 か.F90 .F95 を、固定形式（7 カラム目から記述）なら.f か.F を、2003 形式であれば.f03 か.F03 を付けます。（拡張子が大文字で始まるものは、コンパイルの前に fpp によるプリプロセス処理が行われます。）

コンパイルが成功すると出力ファイル名を指定しない場合は、カレントディレクトリに実行モジュールの a.out が作成されます。

表 3　Fotran,C/C++ コンパイラの主なオプション

| コンパイルオプション | 機能 |
|---|---|
| -On | 最適化レベルを指定する |
| n に指定できる値 | |
| 4 | 言語仕様を逸脱した副作用を伴う最大限の最適化・自動ベクトル化を適用する |
| 3 | 副作用を伴う最適化・自動ベクトル化、および、多重ループの最適化を適用する |
| 2 | （既定値）副作用を伴う最適化・自動ベクトル化を適用する |
| 1 | 副作用を伴わない最適化・自動ベクトル化を適用する |
| 0 | 最適化、自動ベクトル化、並列化、インライン展開を適用しない<br>（最適化レベルを高くすると、最適化の副作用により計算結果が変わることが<br>ありますのでご注意ください） |
| -finline-functions | 自動インライン展開機能を適用する |
| -mparallel | 自動並列化機能を利用する |
| -fopenmp | OpenMP 並列化を利用する |
| -mno-parallel-omp-routine | OpenMP ディレクティブを含むルーチンを自動並列化しない<br>（-mparallel と-fopenmp が同時に指定されたとき、ループが OpenMP の並列区間の<br>外側にある場合は外側のループが自動並列化の対象となります） |
| -ftrace | 性能解析 ftrace 機能用の実行ファイルを作成する |
| -report-diagnostics | ベクトル診断リストを出力する |
| -fdiag-vector=2 | 詳細なベクトル化診断メッセージを出力する（既定値は 1） |
| -report-format | ベクトル化、並列化などの最適化情報がソース行とともに出力された編集リストを<br>出力する |
| -report-all | コード生成リスト、診断メッセージリスト、編集リスト、インラインリスト、<br>オプション リスト、ベクトルリストを出力する |
| -fcheck=bounds | バウンスチェック（配列の上下限のチェック）を行う |
| -fcheck=all | 仮引数のエリアスへの代入、ビット intrinsic な引数、配列の上下限、<br>不正なポインタ、DO ループのステップ値が 0 かどうか、整数オーバーフロー、<br>ポインタ参照、省略可能な引数の参照、不正な再帰呼出しのチェックを行う |

リスト 11　nfort コマンドの使用例

```
（逐次実行）
(sfront)$ nfort コンパイルオプション Fortranソースファイル名

（自動並列化実行）
(sfront)$ nfort -mparallel コンパイルオプション Fortranソースファイル名

（OpenMP並列実行）
(sfront)$ nfort -fopenmp コンパイルオプション Fortranソースファイル名
```

リスト 12　mpinfort コマンドの使用例

```
（MPI並列実行）
(sfront)$ mpinfort コンパイルオプション Fortranソースファイル名

（MPIと自動並列化の同時並列実行）
(sfront)$ mpinfort -mparallel コンパイルオプション Fortranソースファイル名

（MPIとOpenMPの同時並列実行）
(sfront)$ mpinfort -fopenmp コンパイルオプション Fortranソースファイル名
```

**■C/C++ プログラム**　ソースコードは ncc または mpincc コマンドで C プログラムを、nc++ または mpinc++ コマンドで C++ プログラムをコンパイルします。必要に応じて Fortran コンパイラと同じく、表 3 に示したコンパイルオプションをコンパイル時に指定します。その他のコンパイルオプションについては、6 章のコンパイラユーザーズガイドをご参照ください。

ソースファイルの拡張子は、C 言語であれば.c を、C++ であれば.C .cc .cpp .cp .cxx .c++ のいずれかを付けます。

コンパイルが成功すると出力ファイル名を指定しない場合は、カレントディレクトリに実行モジュールの a.out が作成されます。

リスト 13　ncc/nc++ コマンドの使用例

```
（逐次実行）
(sfront)$ ncc コンパイルオプション Cソースファイル名
(sfront)$ nc++ コンパイルオプション C++ソースファイル名

（自動並列化実行）
(sfront)$ ncc -mparallel コンパイルオプション Cソースファイル名
(sfront)$ nc++ -mparallel コンパイルオプション C++ソースファイル名

（OpenMP並列実行）
(sfront)$ ncc -fopenmp コンパイルオプション Cソースファイル名
(sfront)$ nc++ -fopenmp コンパイルオプション C++ソースファイル名
```

リスト 14　mpincc/mpinc++ コマンドの使用例

```
（MPI並列実行）
(sfront)$ mpincc コンパイルオプション Cソースファイル名
(sfront)$ mpinc++ コンパイルオプション C++ソースファイル名

（MPIと自動並列化の同時並列実行）
(sfront)$ mpincc -mparallel コンパイルオプション Cソースファイル名
(sfront)$ mpinc++ -mparallel コンパイルオプション C++ソースファイル名

（MPIとOpenMPの同時並列実行）
(sfront)$ mpincc -fopenmp コンパイルオプション Cソースファイル名
(sfront)$ mpinc++ -fopenmp コンパイルオプション C++ソースファイル名
```

**■実行時性能解析情報（FTRACE）の出力**　コンパイル時に-ftrace オプションを指定すると、実行後にディレクトリに性能解析情報の結果ファイル（ftrace.out）が書き出されます。MPI 並列実行時の性能情報も取得可能です。

性能解析情報の確認は、フロントエンドサーバ上で ftrace コマンドを実行してテキスト形式で確認するか、ftraceviewer コマンドで GUI で確認することができます。図 4 は Ftrace Viewer の表示例です。

FTRACE と Ftrace Viewer についての詳細は、6 章の PROGINF/FTRACE ユーザーズガイド、および NEC Ftrace Viewer ユーザーズガイドをご参照ください。

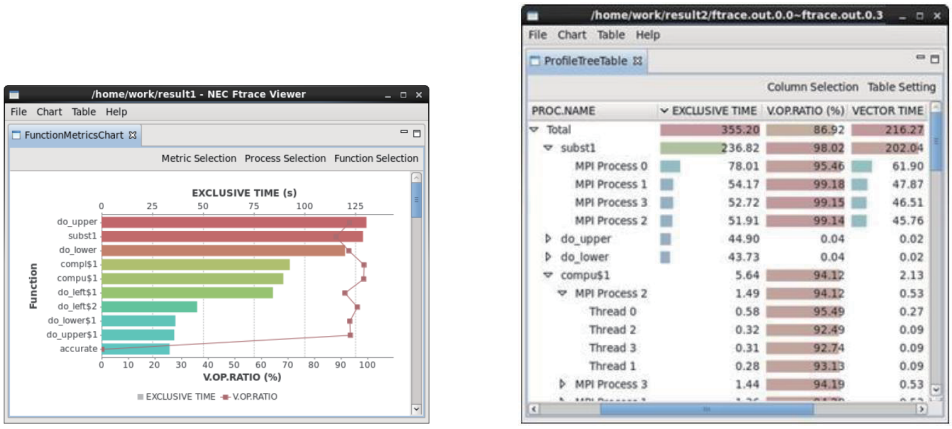


図 4　Ftrace Viewer の表示例

## 4.4　バッチリクエストによるジョブの実行

フロントエンドサーバでコンパイル作業を行って作成したプログラムの実行は、バッチ処理と呼ばれる方法で計算機に実行を依頼します。本センターではバッチ処理に NEC Network Queuing System V（以下、NQSV）を採用しています。

NQSV についてのコマンドやオプションについての詳細は、「NQSV 利用の手引 操作編」をご参照ください。なお、当センター独自の運用方法のためマニュアルの記載の通りに動作しない場合もあります。

### 4.4.1　バッチリクエストの概要

バッチリクエストは、バッチ処理で計算機にジョブの実行を依頼するリクエストのことです。ジョブとは、実行可能ファイル、プログラム、実行モジュールまたはコマンドのことで、リクエストは 1 つまたは複数のジョブから構成されます。

図 5 にバッチリクエスト実行の概念図を示します。

1. ジョブスクリプトの作成 は 4.4.3 から 4.4.5 節で
2. バッチリクエストの投入 は 4.4.6 節で
3. 実行待ち→実行 は 4.4.7 から 4.4.9 節で
4. 実行終了 は 4.4.10 でそれぞれ解説します。

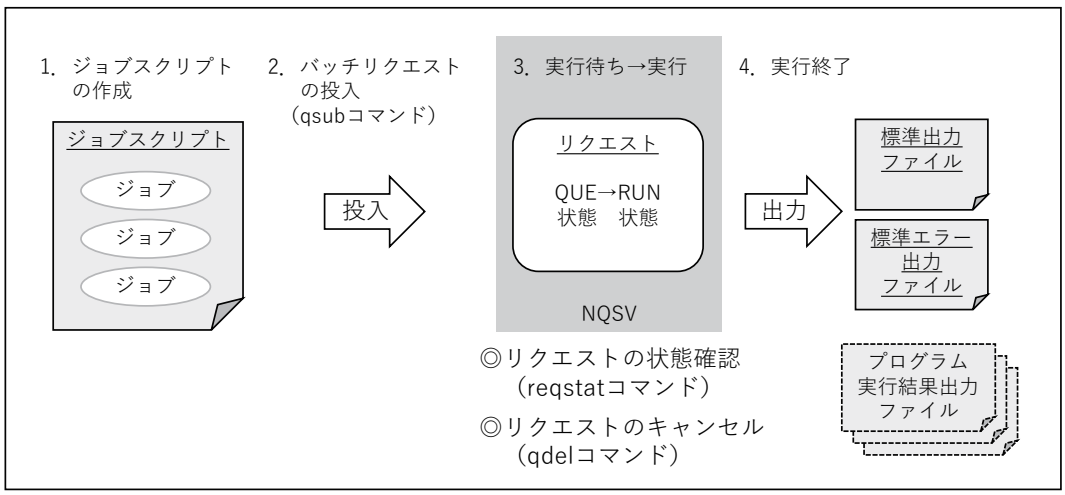


図 5　バッチリクエストの概念図

1. ジョブスクリプトの作成

プログラムの実行手続きを書いたテキストファイルを作成します。利用形態、利用 VE 数、最大経過時間も記述します。ノードの空き状況によっては最大経過時間をデフォルトの時間よりも短く指定することで、バックフィルスケージューリングにより実行開始予定時刻が早くなることがあります。

2. バッチリクエストの投入

3. 実行待ち→実行

以下のコマンドでバッチリクエストを操作します。

- qsub コマンド　　NQSV にバッチリクエストの投入
- reqstat コマンド　投入したリクエストの状態を表示（状態確認）
- qdel コマンド　　投入したリクエストのキャンセル

4. 実行終了

リクエストの実行が終了すると、標準出力と標準エラー出力がファイルとして書き出されます。標準出力/標準エラー出力のファイルサイズ上限は 1GB です。ファイルサイズが上限値を超えた場合はプログラムが強制終了しますので、実行結果はファイル名を指定して書き出しを行ってください。

終了したリクエストは reqstat コマンドの表示から削除されます。

#### 4.4.2　キュー構成

リクエストの投入キューについて説明します。

表 4 に AOBA-S のキュー構成を示します。ジョブスクリプトの冒頭で、表に示したキュー名、利用 VE 数、最大経過時間を指定します。

表 4　サブシステム AOBA-S のキュー構成

| 利用形態 | キュー名 | VE 数 | 最大経過時間<br>規定値/最大値 | メモリサイズ |
|---|---|---|---|---|
| 無料 | sxsf | 1 | 1 時間/1 時間 | 96GB |
| 共有 | sxs | 1～2,048 | 72 時間/720 時間 | 96GB × VE 数 |
| 占有 | 個別設定 | | | 96GB × VE 数 |

経過時間は、バッチリクエストが開始してから終了するまでの時間です。指定した最大経過時間を超えた場合、プログラムの実行中でもバッチリクエストは強制的に終了します。I/O が高負荷となる場合、経過時間が想定よりも遅くなる場合がありますので、必要十分な最大経過時間を指定してください。

なお、最大経過時間は最大値 720 時間（720:00:00）を超えて指定することは出来ません。（sxf では経過時間の最大値は 1:00:00 です。）

#### 4.4.3　バッチリクエストの作成

バッチリクエスト用のシェルスクリプトファイル（ジョブスクリプト）を作成します。通常のシェルスクリプトと同様に、テキストファイル形式で任意のコマンドを組み合わせてプログラムの実行手続きを記述します。ジョブスクリプトを実行するシェルは、sh、csh ともに使用できますが、解説では sh スクリプト形式で記述し、ファイル名を run.sh とします。

ジョブスクリプトに記述する基本項目は以下の通りです。その他に、環境変数の指定やファイル操作コマンドが必要な場合は、適切な箇所に記述します。

- 投入キュー名
- 利用する VE 数
- 最大経過時間
- 作業ディレクトリへの移動コマンド
- プログラムの実行コマンド

#### 4.4.4 ジョブスクリプトの例

リスト 15〜リスト 20 に AOBA-S 向けのジョブのスクリプトファイル例を示します。

**■逐次実行の場合** リスト 15 およびリスト 16 に、逐次実行の場合のジョブスクリプトの例を示します。逐次実行の場合、プログラムは 1VE 内の 1 コアで実行されます。

リスト 15 ジョブスクリプトの例（逐次実行）

```
#!/bin/sh
#PBS -q sxs                     # AOBA-Sを使用する
#PBS --venode 1                 # VEを1個使う
#PBS -l elapstim_req=2:00:00    # 最大経過時間を2時間に指定

cd $PBS_O_WORKDIR               # qsubを実行したディレクトリに移動
./a.out                         # カレントディレクトリの a.out を実行
```

リスト 16 ジョブスクリプトの例（逐次実行、無料キュー）

```
#!/bin/sh
#PBS -q sxsf                    # AOBA-Sの無料キューを使用する
#PBS --venode 1                 # VEを1個使う
#PBS -l elapstim_req=0:30:00    # 最大経過時間を30分に指定

cd $PBS_O_WORKDIR               # qsubを実行したディレクトリに移動
./a.out                         # カレントディレクトリの a.out を実行
```

**■自動並列化/OpenMP 並列実行の場合** リスト 17 に自動並列化/OpenMP 並列実行の場合のジョブスクリプトの例を示します。自動並列化/OpenMP 並列実行の場合、プログラムは 1VE 中の 2〜16 コアで実行されます。実行コア数の指定は環境変数 VE_OMP_NUM_THREADS で行います。逐次実行と同様に、sxf も指定できます。

リスト 17 ジョブスクリプトの例（自動並列化/OpenMP 並列実行）

```
#!/bin/sh
#PBS -q sxs                     # AOBA-Sを使用する
#PBS --venode 1                 # VEを1個使う
#PBS -l elapstim_req=2:00:00    # 最大経過時間を2時間に指定
#PBS -v VE_OMP_NUM_THREADS=16   # 16コア並列で実行

cd $PBS_O_WORKDIR               # qsubを実行したディレクトリに移動
./a.out                         # カレントディレクトリの a.out を実行
```

**■MPI 並列実行の場合** リスト 18 に MPI 並列実行の場合のジョブスクリプトの例を示します。MPI 並列実行の場合は、mpirun コマンドでプログラムの実行を行います。

MPI プロセス数が、確保した VE の物理コア数（VE 数× 16）を超えると演算性能が著しく低下しますのでご注意ください。

逐次実行や自動並列化/OpenMP 並列実行用でコンパイルされたプログラムは、mpirun コマンドで実行を行っても複数 VE での実行はされません。

リスト 18　ジョブスクリプトの例（MPI 並列実行）

```
#!/bin/sh
#PBS -q sxs                     # AOBA-Sを使用する
#PBS --venode 8                 # VEを8個使う
#PBS -l elapstim_req=2:00:00    # 最大経過時間を2時間に指定

cd $PBS_O_WORKDIR               # qsubを実行したディレクトリに移動
mpirun -np 128 ./a.out          # カレントディレクトリの a.out を128プロセス並列で実行
```

**■MPI と自動並列化/OpenMP 同時並列実行の場合**　リスト 19 に MPI と自動並列化/OpenMP の同時並列実行の場合のジョブスクリプトの例を示します。

（MPI プロセス数）×（VE 内並列数）が、確保した VE の物理コア数（VE 数× 16）を超えると演算性能が著しく低下しますのでご注意ください。

リスト 19　ジョブスクリプトの例（MPI と自動並列化／ OpenMP 同時並列実行）

```
#!/bin/sh
#PBS -q sxs                     # AOBA-Sを使用する
#PBS --venode 8                 # VEを8個使う
#PBS -l elapstim_req=2:00:00    # 最大経過時間を2時間に指定
#PBS -v VE_OMP_NUM_THREADS=16   # VE内は16コア並列で実行

cd $PBS_O_WORKDIR               # qsubを実行したディレクトリに移動
mpirun -np 8 ./a.out          # カレントディレクトリの a.out を8プロセス×16コア並列で実行
```

**■標準出力、標準エラーファイルをプロセス毎に分割出力する**　標準出力および標準エラー出力は、1 つのファイルに各プロセスからの出力が混在して書き出されます。このとき、システムで用意されたシェルスクリプト mpisep.sh を利用すると、同一ファイルにプロセス毎の出力が入り混じって出力されないように、MPI プロセスごとにファイルを分けて出力することができます。リスト 20 に示したように、シェルスクリプト/opt/nec/ve/bin/mpisep.sh を実行形式ファイル a.out の前に記述し、環境変数 NMPI_SEPSELECT に 1 から 4 の値を指定することで、表 5 に示した動作を選択できます。

リスト 20　出力をプロセス毎に分割する方法

```
#!/bin/sh
#PBS -q sxs                     # AOBA-Sを使用する
#PBS --venode 8                 # VEを8個使う
#PBS -l elapstim_req=2:00:00    # 最大経過時間を2時間に指定
#PBS -v NMPI_SEPSELECT=3        # 出力形式3を指定

cd $PBS_O_WORKDIR               # qsubを実行したディレクトリに移動
mpirun -np 128 /opt/nec/ve/bin/mpisep.sh ./a.out
                                # カレントディレクトリの a.out を128プロセス並列で実行
```

表 5　NMPI_SEPSELECT の引数と出力形式

| NMPI_SEPSELECT の引数 | 出力形式 |
|---|---|
| 1 | MPI プロセスの標準出力をプロセス別のファイルに保存 |
| 2 | （既定値）MPI プロセスの標準エラー出力を<br>プロセス別のファイルに保存 |
| 3 | MPI プロセスの標準出力および標準エラー出力を<br>それぞれプロセス別のファイルに保存 |
| 4 | MPI プロセスの標準出力および標準エラー出力を<br>プロセス別の 1 つのファイルに保存 |

実行時に stdout.*や stderr.*の同名ファイルが存在する場合は、上書きではなくファイルに追記します。

### 4.4.5 qsub コマンドの主なオプション

表 6 に、qsub コマンドの主なオプションと機能を示します。ジョブスクリプトの冒頭から、#PBS の後に各オプションを記述します。

表 6 qsub コマンドの主なオプションと機能

| オプション | 機能 | 指定 |
|---|---|---|
| -q 投入キュー名 | 投入キュー名を指定 | 必須 |
| –venode 利用 VE 数 | AOBA-S で利用する VE 数を指定<br>（このオプションは先頭のハイフンが 2 つ） | 必須 |
| -l elapstim_req=hh:mm:ss | 最大経過時間を指定 | 必須（注 1 参照） |
| -A 課金先番号（PJ） | 課金先の番号を指定 | 任意（注 2 参照）<br>省略時：デフォルトの PJ |
| -N リクエスト名 | リクエスト名を指定 | 任意<br>省略時：ジョブスクリプトファイル名 |
| -o ファイル名 | 標準出力のファイル名を指定 | 任意　省略時：<br>リクエスト名.o リクエスト ID |
| -e ファイル名 | 標準エラー出力のファイル名を指定 | 任意　省略時：<br>リクエスト名.e リクエスト ID |
| -jo | 標準出力および標準エラー出力を<br>同一ファイルに出力 | 任意 |
| -m b | リクエスト実行開始時にメールを送信 | 任意 |
| -m e | リクエスト実行終了時にメールを送信 | 任意 |
| -m be | リクエスト実行開始時と終了時にメールを送信 | 任意 |
| -M メールアドレス | メールの送信先を指定 | 任意 |
| -r y または n | リクエストのリラン可否を指定（注 3 参照） | 任意　省略時：y |

**（注 1）** 経過時間はバッチリクエストが実行を開始してから、終了するまでの時間です。プログラムの実行に、最大でどれくらいの経過時間の確保が必要かを指定します。指定した最大経過時間が短いリクエストほど計算資源が確保されやすく、実行待ちの時間を短縮することができます。

ただし、指定した最大経過時間を超えると、実行は打ちきりとなり強制終了します。I/O が高負荷となる場合、経過時間が想定よりも長くなることがありますので、必要十分な時間を指定してください。

**（注 2）** デフォルトの課金先番号や、利用可能な課金先番号の確認は、フロントエンドサーバ上で project コマンドを実行します。デフォルトの課金先番号の変更も可能です。

**（注 3）** リランとは、リクエストを始めから実行し直すことで、計算機のメンテナンスや障害発生時に管理者側でリクエストをリランする場合があります。

リランによりプログラムの実行に不都合が生じる場合（実時間で時間計測を行っている場合など）は、-r n（リランしない）を指定してください。

### 4.4.6 バッチリクエストの投入

バッチリクエストの投入は qsub コマンドで行います。リスト 21 に qsub コマンドの実行例を示します。ジョブスクリプトファイル名が run.sh の場合です。

リスト 21 qsub コマンドによるバッチリクエスト投入

```
(sfront)$ qsub run.sh
```

バッチリクエストが正常に投入されるとリスト 22 のようなメッセージが表示されます。この例では、リクエスト ID には 1234.sjob が割り当てられ、投入先は sxs キュー、課金先番号は un0000 が指定されたことを示しています。リクエスト ID はバッチリクエストの状況確認やキャンセルの際に用います。

バッチリクエストの投入が失敗した場合は、エラーメッセージが表示されますので、ジョブスクリプトの記述などに誤りがないかを確認してください。

リスト 22　qsub コマンドによるバッチリクエスト投入

```
(sfront)$ qsub run.sh
プロジェクトコード:un0000 にリクエストを投入します
Request 1234.sjob submitted to queue : sxs.
```

#### 4.4.7　バッチリクエストの実行

投入されたリクエストは、実行待ちの列に並びます。リクエストの実行順序はバックフィルスケージューリングで制御されます。

バックフィルスケージューリングでは、計算資源が確保できたリクエストから実行を開始します。実行に必要な計算資源量（利用 VE 数×最大経過時間）が少ないリクエストは、投入順序によらず早く実行開始する場合があります。

#### 4.4.8　バッチリクエストの状態確認

バッチリクエストの状態確認は reqstat コマンドで行います。reqstat コマンドを実行すると、実行待ちおよび実行中のリクエストの状態が表示されます。該当するリクエストがない場合は何も表示されません。

表 7 に reqstat コマンドの表示項目を、表 8 にリクエストの状態の説明を示します。

表 7　reqstat コマンドの表示項目

| 表示項目 | 内容 |
|---|---|
| RequestID | リクエスト ID |
| RequestName | リクエスト名 |
| User | 利用者番号 |
| PJCode | 課金先番号（プロジェクトコード） |
| Que | 実行キュー名 |
| Node | 実行時に指定した VE 数 |
| ElapseLimit | 投入時に指定した最大経過時間 |
| STT | リクエストの状態 |
| StartTime | 実行開始予定時刻<br>（実行開始後は実際の実行開始時刻） |
| Memory1 | 現在の VH 使用メモリ量 |
| Memory2 | 現在の VE 使用メモリ量 |
| ElapseTime | 現在までの経過時間 |
| NodeTime | 現在までのノード時間（ElapseTime × Node）<br>（課金対象時間） |

表 8　reqstat コマンドの表示項目

| 表記 | リクエストの状況 |
|---|---|
| RUN | 実行中 |
| QUE | 実行待ち |
| EXT | 標準出力/標準エラー出力ファイルの出力中（※） |

リクエストの実行開始予定時刻は、StartTime の欄に表示されます。実行開始予定時刻は他のリクエストの状況により随時更新されます。通常は更新前の時刻よりも遅くなることはありませんが、システムの障害発生時にはその限りではありません。
※　以下のような場合、リクエストが EXT 状態で長く停滞することがあります。他の利用者のリクエストの実行に影響が出ますのでご注意ください。

- ファイル容量がクォータ値を超えたため、標準出力/標準エラー出力ファイルを出力できない
  ファイルを整理して容量を空けてください。空き容量が確保されるまで EXT 状態で停滞したままになります。容量が確保され次第、ファイルが出力されリクエストが終了します。
- システム上で問題が発生している
  管理者で対応しますのでそのままお待ちください。状況をメールでご連絡いたします。

ファイル容量の追加方法は以下のページをご参照ください。
【データ転送（ストレージ）】https://www.ss.cc.tohoku.ac.jp/storage/

#### 4.4.9　バッチリクエストのキャンセル

投入したリクエストをキャンセルする場合は、qdel コマンドを使用します。qdel コマンドを実行すると実行中のプログラムは強制終了し、リクエストは削除されます。リクエストのキャンセルは投入した利用者番号から行うことができ、他利用者のリクエストはキャンセルできません。

リスト 23 に qdel コマンドの実行例を示します。1234.sjob というジョブ ID のリクエストをキャンセルする例で、qdel に続けてキャンセルするリクエスト ID を指定します。リクエスト投入時、または reqstat コマンドで表示されるリクエスト ID を指定してください。

リクエストがキャンセルされるとメッセージが表示されます。

リスト 23　qdel コマンド

```
(sfront)$ qdel 1234.sjob
Request 1234.sjob was deleted.
```

#### 4.4.10　バッチリクエストの終了

リクエストが終了すると、reqstat コマンドで表示されなくなります。終了したリクエストの実行結果ファイルとして、リクエスト実行中に標準出力された内容を保存した「標準出力ファイル」と、標準エラー出力の内容を保存した「標準エラー出力ファイル」が作成されます。

ジョブのスクリプトでオプション -o と-e を指定した場合はそのファイル名で作成されます。指定しない場合は以下のファイル名で作成されます。

- 標準出力ファイル：　リクエスト名.o リクエスト ID
- 標準エラーファイル：　リクエスト名.e リクエスト ID

なお、標準出力/標準エラー出力のファイルサイズの上限値は 1GB です。ファイルサイズが上限値を超えた場合はプログラムが強制終了しますので、実行結果はプログラム内でファイル名を指定して書き

出すようにしてください。

## 4.5　会話リクエストの投入と利用

AOBA-S では、qlogin コマンドを利用したセッション接続タイプの会話リクエストの投入が可能です。会話リクエストでは 1VH+8VE を利用して、ソースコードのコンパイル、会話型形式での実行、および GDB（ve-gdb）の利用が可能です。

会話リクエストはバッチリクエストと同様に演算負担額が発生します。会話リクエストのセッションが開始されてからセッションが切断されるまで、が演算負担額の対象となります。ジョブの実行を行っていない場合でも、セッションの接続時間が課金対象時間となりますのでご注意ください。

会話リクエストの最大経過時間は 1 時間です。会話リクエスト用ノードの利用状況により、セッションの開始まで実行待ちが発生することがあります。

### 4.5.1　qlogin による会話リクエストの投入方法

会話リクエストの投入は、AOBA-S 用フロントエンドサーバ上で qlogin コマンドで行います。リスト 24 に qlogin コマンドの例を示します。

リスト 24　qlogin コマンド

```
(sfront)$ qlogin -q inter -l elapstim_req=1800
                              # 会話キューinterの指定（必須）と最大経過時間の指定（任意）
プロジェクトコード：un0000 にリクエストを投入します
Request 1234.sjob submitted to que: inter. # 会話リクエストが受け付けられた
Wating for 1234.sjob to start.             # 実行ホストとのセッション接続待ち
(sxat3001)$                                # 実行ホストのシェルプロンプトに表示が変わる
(sxat3001)$ exit                           # セッションの切断
(sfront)$                                  # シェルプロンプト表示が戻る
```

リスト 25 に ve-gdb（VE 用 GDB）の起動方法を示します。

リスト 25　ve-gdb コマンド

```
(sxat3001)$ ve-gdb a.out
No symbol table is loaded.  Use the "file" command.
GNU gdb (GDB) 7.12.1-8.el8
Modified by NEC Corporation for the VE port, 2017-2019
Modified by Arm. Copyright (C) 2002-2019 Arm Limited (or its affiliates).
All rights reserved.
（省略）
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from a.out...(no debugging symbols found)...done.
(gdb)
```

VE 用の GDB と一般的な GDB の相違点については、6 章の GDB の相違点をご参照ください。

# 5 SX-Aurora TSUBASA 用数値演算ライブラリ

## 5.1 NLC (NEC Numeric Library Collection)

NEC Numeric Library Collection は、広範な分野の数値シミュレーションプログラムの作成を強力に支援する数学ライブラリのコレクションであり、Vector Engine に対応しています。NEC Numeric Library Collection を用いることにより、難解な数値計算アルゴリズムの詳細に煩わされることなく高度な科学技術計算プログラムを作成することができ、数値シミュレーションプログラム開発の生産性を大幅に改善することができます。

NEC Numeric Library Collection は、Fortran または C 言語プログラムから利用できます。

### 5.1.1 Fortran プログラムから利用する場合

Fortran プログラムから利用する場合、表 9 に示すライブラリが使用できます。

表 9 Fortran 用 NLC の機能概要

| ライブラリ名 | | 機能概要 |
|---|---|---|
| ASL | ネイティブインタフェース | 数値計算・統計計算のための各種アルゴリズムを備えた<br>科学技術計算ライブラリ |
| | 統合インタフェース | フーリエ変換、乱数、ソート |
| | FFTW3 インタフェース | FFTW (version 3.x) の API で ASL のフーリエ変換を<br>利用するためのインタフェースライブラリ |
| BLAS | | ベクトル、行列の基本演算 |
| LAPACK | | 連立 1 次方程式、固有値方程式、特異値分解 |
| ScaLAPACK | | 連立 1 次方程式、固有値方程式、特異値分解 (分散メモリ並列用) |
| BLACS | | ベクトル、行列の基本演算のためのメッセージパッシングライブラリ<br>(分散メモリ並列用) |
| SBLAS | | スパース行列の基本演算 |
| HeteroSolver | | 連立 1 次方程式 (スパース行列用の直接法ソルバ) |
| Stencil Code Accelerator | | ステンシル計算の加速 |

### 5.1.2 C プログラムから利用する場合

C プログラムから利用する場合、表 10 に示すライブラリが使用できます。

表 10 C 言語用 NLC の機能概要

| ライブラリ名 | | 機能概要 |
|---|---|---|
| ASL | ネイティブインタフェース | 数値計算・統計計算のための各種アルゴリズムを備えた<br>科学技術計算ライブラリ |
| | 統合インタフェース | フーリエ変換、乱数、ソート |
| | FFTW3 インタフェース | FFTW (version 3.x) の API で ASL のフーリエ変換を<br>利用するためのインタフェースライブラリ |
| CBLAS | | BLAS C 言語インタフェース |
| SBLAS | | スパース行列の基本演算 |
| HeteroSolver | | 連立 1 次方程式 (スパース行列用の直接法ソルバ) |
| Stencil Code Accelerator | | ステンシル計算の加速 |

NLC の詳細、およびコンパイルとリンク方法については 6 章の NLC (NEC Numeric Library Collection) ユーザーズガイドをご参照ください。

# 6　マニュアル

サブシステム AOBA-S についてのマニュアルはオンライン上で公開されています。以下リンク先の NEC Aurora Forum の NEC SX-Aurora TSUBASA Documentation をご参照ください。英語版、日本語版ともに提供されています。

【NEC Aurora Forum Documentation】https://www.hpc.nec/documentation

当センター独自の運用方法により、マニュアルに記載の事項が動作しない場合もありますのでご注意ください。

## 6.1　コンパイラマニュアル

コンパイラについては以下のマニュアルをご参照ください。

■SDK

- C/C++ Compiler ユーザーズガイド、C/C++ Compiler User's Guide
- Fortran Compiler ユーザーズガイド、Fortran Compiler User's Guide
- NEC Parallel Debugger ユーザーズガイド、NEC Parallel Debugger User's Guide

■NEC MPI

- NEC MPI ユーザーズガイド、NEC MPI User's Guide

## 6.2　性能情報取得についてのマニュアル

実行時の性能情報取得については以下のマニュアルをご参照ください。

■SDK

- PROGINF/FTRACE ユーザーズガイド、PROGINF/FTRACE User's Guide
- NEC Ftrace Viewer ユーザーズガイド、NEC Ftrace Viewer User's Guide

## 6.3　NQSV についてのマニュアル

NQSV の利用方法については以下のマニュアルをご参照ください。

■NQSV

- NQSV 利用の手引 操作編、NQSV User's Guide Operation

## 6.4　GDB についてのマニュアル

VE 用の GDB と一般的な Linux の GDB の使用法に関する相違点については以下の資料をご参照ください。

■VEOS

- GDB の相違点

### 6.5 数値計算ライブラリ（NLC）マニュアル

数値計算ライブラリ（NLC）については以下のマニュアルをご参照ください。

■SDK

- NLC (NEC Numeric Library Collection) ユーザーズガイド、
  NLC (NEC Numeric Library Collection) User's Guide

## 7 おわりに

本稿では、スーパーコンピュータ AOBA のサブシステム AOBA-S のプログラミング利用ガイドとして基本的な手順を紹介しました。研究室のサーバでは実現できなかったプログラムやアイデアを、ぜひ最新鋭のスーパーコンピュータでお試しください。研究の強力なツールとしてご活用いただければ幸いです。

ご不明な点、ご質問等がありましたら、お気軽にセンターまでお問い合わせください。お問い合わせについては利用相談フォームをご利用ください。

【利用相談フォーム】https://www.ss.cc.tohoku.ac.jp/consultation/

センターから運用に関するお知らせは以下をご参照ください。

【センターからのお知らせ】https://www.ss.cc.tohoku.ac.jp/information/