東北大学サイバーサイエンスセンター
新スーパーコンピュータAOBAの紹介と利用法

2020年11月5日

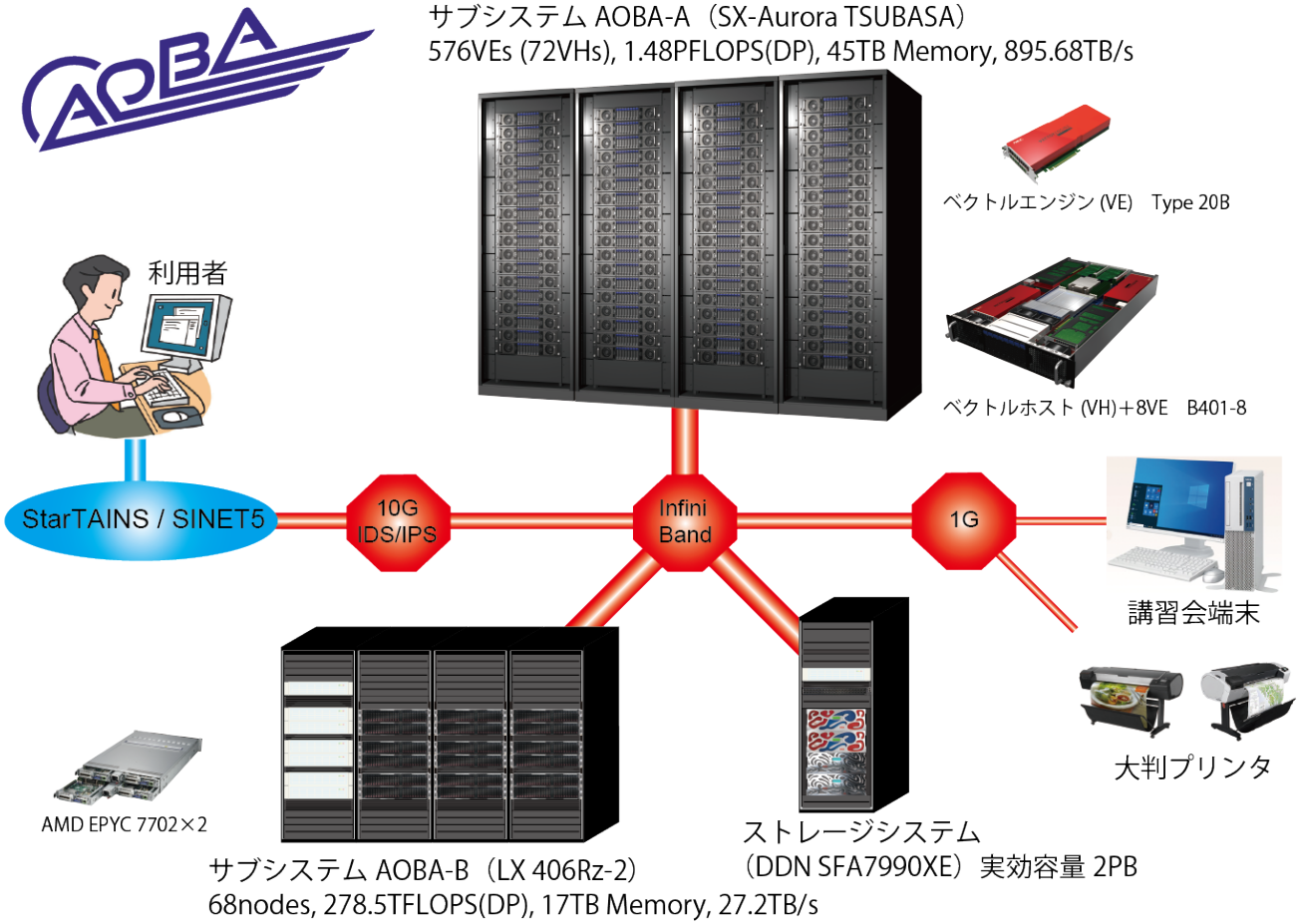
東北大学 サイバーサイエンスセンター
東北大学 情報部 情報基盤課

内容

- スーパーコンピュータAOBAの紹介
- システムの特徴
- AOBA-AとAOBA-Bどちらを使うか？
- 負担金制度
- 利用者番号とプロジェクトコード
- 利用者向けウェブサイトとポータルサイト
- 鍵の作成からログインまで
- プログラムのコンパイル方法
- ジョブの実行方法と実行キュー
- 利用申請方法（利用者番号の取得）
- 利用者講習会・利用相談・高速化支援
- 今後の講習会予定

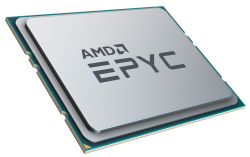
スーパーコンピュータAOBAの紹介

- サブシステムAOBA-A　：SX-Aurora TSUBASA
- サブシステムAOBA-B　：LX 406Rz-2
- ストレージシステム　：DDN SFA7990XE（2PB）
- 利用者向けサーバー　：ログインサーバ，フロントエンドサーバ，ファイル転送サーバ
- センター内施設　：講習会端末，大判プリンタ（A0判）

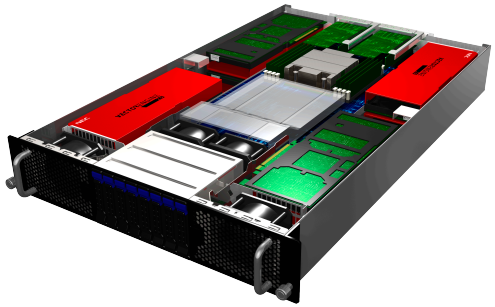


・ベクトルプロセッサ＋x86/Linuxアーキテクチャの構成
　ベクトルエンジン（**VE**）は演算処理を行う
　　→ 本システムでは Type 20B を8個搭載
　ベクトルホスト（**VH**）はOS処理，VE制御，I/O制御等を行う
　　→ 本システムでは EPYC 7402P を1個搭載
　基本システム単位は 1**VH** ＋ 8**VE**
　システム全体では 72**VH**＋576**VE**　最大利用は32VH+256VE
　ノード間接続は InfiniBand HDR（200Gbps）×2

ベクトルエンジン（**VE**）
Type 20B



ベクトルホスト（**VH**）
AMD EPYC 7402P


1**VH**＋8**VE**
B401-8

・**VE**はSX-ACEを継承するベクトルアーキテクチャ
　マルチコアベクトルプロセッサ（8コア）とHBM2メモリを搭載
　ベクトル演算による高い演算性能：**VE**あたり 2.45TFLOPS（DP）4.91TFLOPS（SP）
　コア間共有メモリ：**VE**あたり 48GB
　高いメモリバンド幅：**VE**あたり 1.53TB/s
　演算性能とデータ転送性能のバランス：0.65B/F

・**VH**のLinux OS環境とVEを連携した利用が可能
　24コア 1.075TFLOPS（DP） 256GBメモリを搭載
　【**VH**でプリ処理 → **VE**で演算処理 → **VH**でポスト処理】を1ジョブで完結するなどの利用が可能

・自動ベクトル化・自動並列化機能機能を備えた，Fortran/C/C++コンパイラを利用可能
　x86向けに開発されたFortran/C/C++ソースコードも，コンパイラがベクトル性能を引き出す
　SX-ACE向けに開発されたアプリケーションの移植もサポート
　MPIライブラリによる分散メモリ並列実行に対応
　GNU互換環境を装備（SX-ACE向けコンパイラからオプション，指示行に仕様の変更あり）

・ベクトルアーキテクチャに最適化された科学技術計算ライブラリ（Fortran/C）
　BLAS，FFTW，LAPACK，ScaLAPACKのインターフェースをそのまま利用可能
　ASLインタフェースも利用可能

・実行時性能解析ツールを利用可能
　PROGING，FTRACE，Ftrace Viewer

・AI分野への活用
　ミドルウェア「Frovedis」による統計的機械学習処理の高速化
　「TensorFlow」と「Python」の利用も可能

・一部の量子化学分野のアプリもインストール済
　VASP，Quantum Espressoを**VE**でも利用可能
　その他対応アプリについては https://jpn.nec.com/hpc/sxauroratsubasa/Application/index.html をご参照下さい。

・x86/Linuxアーキテクチャ構成

1ノードにAMD EPYC 7702（64コア）を2個，256GBメモリを搭載

ノード性能は 4.096TFLOPS（DP）8.192TFLOPS（SP）409.6GB/s

システム全体では68ノード　最大利用は16ノード

ノード間接続は InfiniBand HDR（200Gbps）×1

AMD EPYC 7720

・AMDプロセッサに最適化された Fortran/C/C++コンパイラ，科学技術計算ライブラリを利用可能

AOCC (AMD Optimizing C/C++ Compiler)
AMD uProf, AMD Optimizing CPU Libraries
Open MPI

・GNUコンパイラ，Intelコンパイラも利用可能

GNU Compiler Collection(Fortran,C/C++)，OpenMPI

Intel Parallel Studio XE Cluster Edition（ライセンス数限定）

LX 406Rz-2 4ノード

・Linux OSに対応したアプリを準備中

Gaussian16，GRRM17，VASP，Quantum Espresso，OpenFOAM

（利用者限定商用アプリ）Mathematica，MATLAB

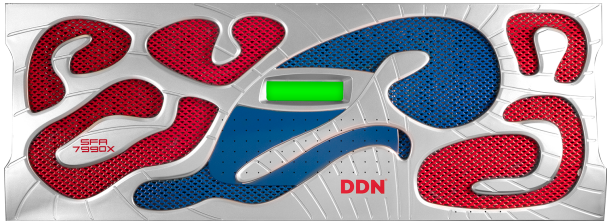
他OSSなどもユーザ領域にインストール可能

・ScaTeFSによる高速な分散・並列ファイルシステム

実効容量 2PB

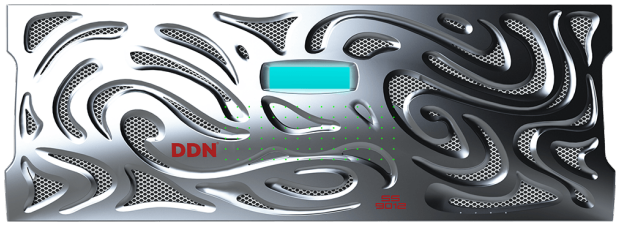
AOBA-A, AOBA-B, フロントエンドサーバおよびファイル転送サーバと接続

接続は InfiniBand HDR（100Gbps）×8



コントローラー部

DDN SFA7790XE



スピンドル格納部

DDN SS9012

システムの特徴（5/5）　ログインサーバ，フロントエンドサーバ，ファイル転送サーバ

・ログインサーバ
　（一般利用者向け）
　　SSH認証方式によるログイン
　　緊急度の高い保守に対応するための踏み台サーバ

ログインサーバ

・フロントエンドサーバ
　（一般利用者向け）
　　ログインサーバからSSH認証方式によるログイン
　　コンパイル，バッチリクエスト投入など
　　GUIアプリケーション（MATLAB，Mathematica）の利用
　（HPCI利用者向け）
　　GSI-SSH認証方式によるログイン
　　コンパイル，バッチリクエスト投入など

フロントエンドサーバ
（一般利用者向け，HPCI利用者向け）
データ転送サーバ

・データ転送サーバ
　（一般利用者，HPCI利用者向け）
　　SSHによるファイル転送

	AOBA-A (SX)	AOBA-B (LX)
利用**最小**単位	1VE	1ノード
コア数	8	**128**
理論演算性能 [TFLOPS]	2.45	**4.09**
メモリ容量 [GB]	48	**256**
コア性能 [GFLOPS]	**307**	32
メモリ転送性能 [TB/s]	**1.53**(8コア)	0.40

	AOBA-A (SX)	AOBA-B (LX)
利用**最大**単位	256VE	16ノード
コア数	2,048	2,048
理論演算性能 [TFLOPS]	**627.2**	65.5
メモリ容量 [TB]	**12**	4

AOBA-Aを選択

・シングルコア実行のプログラム
・メモリ転送性能が律速になるプログラム
・MPIによる大規模並列を行う場合

AOBA-Bを選択

・ノード内並列化（OpenMP並列・自動並列）されているプログラム
・ノード内でメモリ容量を多く使うプログラム

AOBA-Aを選択

・SX-ACEで利用していたプログラム
・ベクトル化率が99.0%以上，平均ベクトル長が128以上のプログラム
　→ 実行時性能の取得方法と，さらなるベクトル高速化については利用相談をご利用下さい。
・お試しで使いたい
　→ IVEを1時間まで利用できる無料のキューがあります。（同時実行数1）

AOBA-Bを選択

・商用アプリ，OSSなどソースコードを改変しにくいプログラム
・ベクトル性能が出ないと分かっているプログラム
　→ 性能調査とベクトル高速化については利用相談をご利用下さい。
・AOBA-A向けにコンパイル出来ないプログラム
・Fortran/C/C++以外のコンパイラを使用するプログラム
・Gaussian16，GRRM17，MATLAB（バッチ処理），OpenFOAMを使う場合

両方で実行を試した後に選択

・AOBA-A，AOBA-Bの両方で実行できるアプリケーション（VASP5.4，Quantum Espresso6.3のpw.x）

負担金制度（1/3）

・利用者番号（アカウント）の初期登録料，年間維持費**なし**

- 従量課金を基本とするため，計算機を利用しない場合の負担金請求は**0円**

※ 利用者番号は年度を超える場合も自動継続され，ホーム領域（uhome）のデータも保存されます。

・計算機利用負担金

- 共有利用・従量

  課金対象時間（利用VH数または利用ノード数と，利用時間の積）に比例した課金方式

- 共有利用・定額

  利用負担金の先払いにより，負担額の課金対象時間相当まで計算機を利用可能

  年度途中に定額負担金の追加も可能

- 占有利用

  3ヶ月単位でAOBA-A（8VE単位）またはAOBA-B（1ノード単位）を占有して利用

  特定利用者で計算資源を占有するため，他利用者のジョブ待ちが無い

・ストレージ負担経費

- ホーム領域 5TBまで無料
- 追加1TBにつき年額3,000円

・出力負担経費

- センターの大判プリンタ1枚につき ソフトクロス紙 1,200円　光沢紙 600円

・民間企業利用については 成果公開型は2倍，成果非公開型は4倍の課金単価

負担金制度（2/3）

サブシステムAOBA-A（SX-Aurora TSUBASA）

利用形態	負担額および利用可能課金対象時間
共有 （無料）	利用 **VE 数 1**（実行数，実行時間の制限あり）**無料**
共有 （従量）	課金対象時間 ＝（利用 VE 数÷ 8 を切り上げた数）×経過時間（秒） 課金対象時間の合計 1 時間につき **125 円** 課金対象時間は半期毎 (4〜9 月および 10〜3 月) に合計し，1 時間未満を切上げて負担金を請求する 。
共有 （定額）	負担額 **10 万円**につき 課金対象時間の合計 **800時間**
占有	利用**VE数8**，利用期間**3ヶ月**につき **270,000円**

サブシステムAOBA-B（LX 406-Rz2）

利用形態	負担額および利用可能課金対象時間
共有 （従量）	課金対象時間 ＝ 利用ノード数 × 経過時間（秒） 課金対象時間の合計 1 時間につき **22 円** 課金対象時間は半期毎 (4〜9 月および 10〜3 月) に合計し，1 時間未満を切上げて負担金を請求する 。
共有 （定額）	負担額 **10 万円**につき 課金対象時間の合計 **4,600時間**
占有	利用**ノード数1**，利用期間**3ヶ月**につき **47,000円**

・民間企業利用については 成果公開型は**2倍**，成果非公開型は**4倍**の課金単価

サブシステムAOBA-A（SX）

課金対象時間＝10,000【円】÷125【時間単価】＝ 80【時間】

1VE～8VEまで同じ時間単価なので,

理論演算性能（DP）2.45TFLOPS～19.60TFLPOS

物理メモリ 48GB～384GB

を80時間分利用できる

サブシステムAOBA-B（LX）

課金対象時間＝10,000【円】÷22【時間単価】≒ 454【時間】

1ノードで利用した場合は

理論演算性能（DP）4.096TFLOPS　物理メモリ 256GB

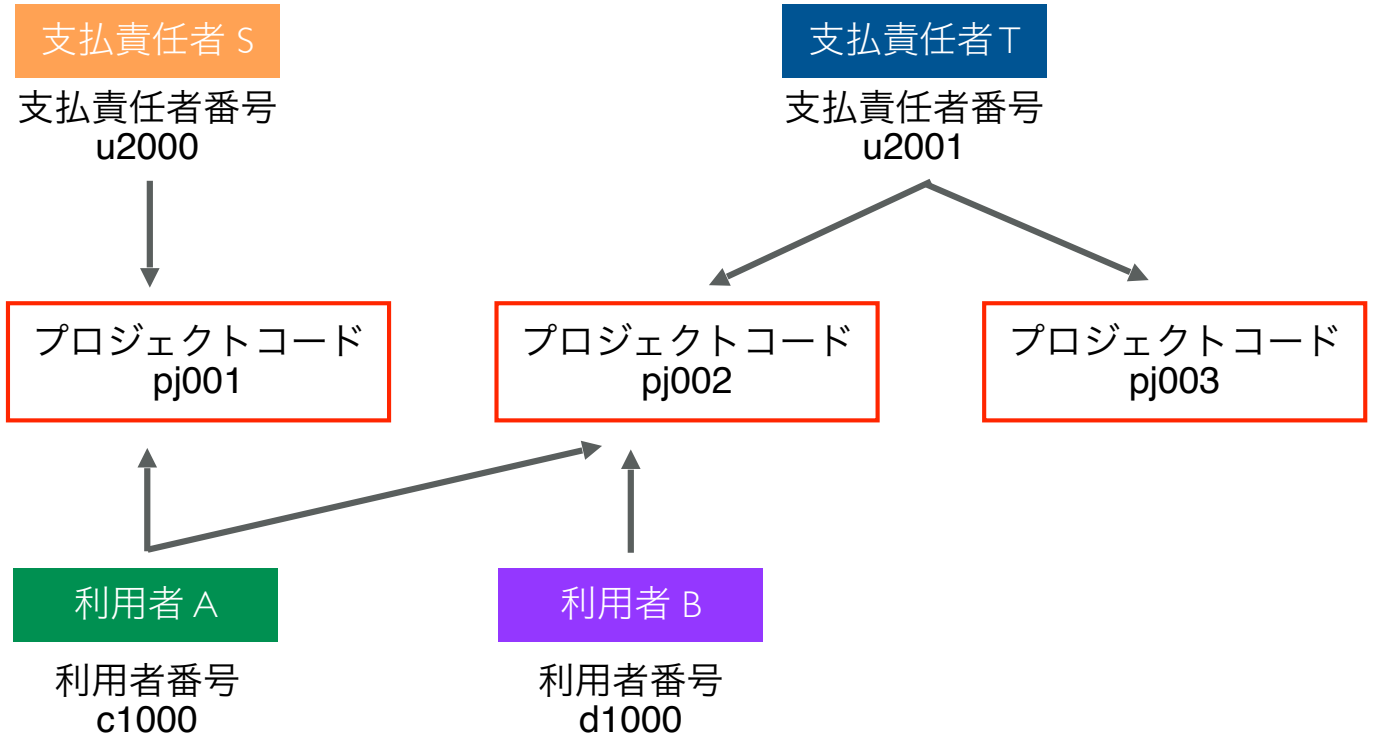
を約454時間分利用できる

16ノードで利用した場合は

理論演算性能（DP）65.5TFLOPS　物理メモリ 4TB

を約28時間分利用できる

利用者番号とプロジェクトコード



1つの利用者番号から，複数のプロジェクトコードに対してジョブを投入

→ 利用者環境を切り替えずに，課金先（請求先）の使い分けが可能

プロジェクトコードによる予算管理

→ 投入可能リソース，課題利用期間の管理が容易

利用者向けウェブサイトとポータルサイト

・サイバーサイエンスセンター 大規模科学計算システムのウェブサイト　https://www.ss.cc.tohoku.ac.jp/
  - システムの利用マニュアル
  - 運用についてのお知らせ
  - 利用相談などの連絡先
  - 講習会予定

・利用申請からログインまでについては以下を参照

https://www.ss.cc.tohoku.ac.jp/first-use/

・利用者用ポータルサイト（LDAP認証連携）
  - 公開鍵・秘密鍵ペアの作成
  - 利用状況（負担金，合計課金対象時間，ジャーナルレコード等）の確認
    従来のフロントエンドサーバ上のコマンドでも確認可能

鍵の作成からログインまで

ログインサーバ
login.cc.tohoku.ac.jp

・フロントエンドサーバへのログイン

フロントエンドサーバ

・プログラムのコンパイル
・バッチリクエストの投入
（AOBA-A，AOBA-B での実行）
・バッチリクエストの確認、削除
・結果の確認

データ転送サーバ
file.cc.tohoku.ac.jp

・ストレージシステムとのデータ転送

・既存の利用者番号の鍵ペアは継続して利用可能
・新規利用者と鍵の再作成が必要な場合
  ▫ ポータルサイトに利用者番号と初期パスワードでログインして鍵ペアを作成
・一般利用者はログインサーバを経由してフロントエンドサーバにログイン
    → セキュリティインシデントに即時対応するため
  ▫ HPCI利用者（GSI-SSH認証）は hfront.cc.tohoku.ac.jp を利用
・ローカルPCとのデータ転送は，データ転送サーバを利用
・これらサーバでのプログラム実行は制限されるので、プログラムはバッチリクエストで実行

NEC Software Development kit for Vector Engine

・逐次実行

front$ **nfort** コンパイルオプション Fortranソースファイル名

front$ **ncc** コンパイルオプション Cソースファイル名

front$ **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

front$ **nfort -mparallel** コンパイルオプション Fortranソースファイル名

front$ **ncc -mparallel** コンパイルオプション Cソースファイル名

front$ **nc++ -mparallel** コンパイルオプション C++ソースファイル名

・OpenMP並列化

front$ **nfort -fopenmp** コンパイルオプション Fortranソースファイル名

front$ **ncc -fopenmp** コンパイルオプション Cソースファイル名

front$ **nc++ -fopenmp** コンパイルオプション C++ソースファイル名

・MPI並列化（自動並列化，OpenMP並列化の併用も可能）

front$ **mpinfort** コンパイルオプション Fortranソースファイル名

front$ **mpincc** コンパイルオプション Cソースファイル名

front$ **mpinc++** コンパイルオプション C++ソースファイル名

AMD Optimizing C/C++ Compiler （AOCC）

・逐次実行

front$ **flang**コンパイルオプション Fortranソースファイル名
front$ **clang** コンパイルオプション Cソースファイル名
front$ **clang++** コンパイルオプション C++ソースファイル名

・OpenMP並列化

front$ **flang -fopenmp** コンパイルオプション Fortranソースファイル名
front$ **clang -fopenmp** コンパイルオプション Cソースファイル名
front$ **clang++ -fopenmp** コンパイルオプション C++ソースファイル名

・MPI並列化（OpenMPIを利用，OpenMP並列化の併用も可能）

front$ **mpifort** コンパイルオプション Fortranソースファイル名
front$ **mpicc** コンパイルオプション Cソースファイル名
front$ **mpic++** コンパイルオプション C++ソースファイル名

最適化のコンパイルオプション　-march=znver2 でRome向け最適化コンパイル

・既存コードの移行用として，**Intel Compiler, IntelMKL, Intel MPI**をライセンス数限定で提供

バッチリクエストの実行方法と実行キュー（1/5）

・ジョブスクリプトファイルを作成し，qsub コマンドでリクエストを投入

front$ **qsub** ジョブスクリプトファイル名

・バッチリクエストの実行状況，リクエストIDは，reqstat コマンドで確認

front$ **reqstat**

・バッチリクエストのキャンセル，途中終了はqdelコマンド

front$ **qdel** リクエストID

【逐次実行】

```
#!/bin/sh                          #シェルを指定
#PBS -q sx --venode 1              #SX-Auroraを1VE使用する
#PBS -l elapstim_req=2:00:00       #実行時間を2時間に設定
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／OpenMP並列実行】

```
#!/bin/sh                          #シェルを指定
#PBS -q sx --venode 1              #SX-Auroraを1VE使用する
#PBS -l elapstim_req=2:00:00       #実行時間を2時間に設定
#PBS -v OMP_NUM_THREADS=8          #1VEあたり8コアで実行（1～8）
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実行】

```
#!/bin/sh                          #シェルを指定
#PBS -q sx --venode 8              #SX-Auroraを8VE使用する
#PBS -l elapstim_req=2:00:00       #実行時間を2時間に設定
cd $PBS _O_WORKDIR                 # qsubを実行したディレクトリに移動
mpirun -np 64 ./a.out              #カレントディレクトリのa.outを64プロセスで実行
```

【MPIと自動並列／OpenMP並列実行の同時利用】

```
#!/bin/sh                          #シェルを指定
#PBS -q sx --venode 8              #SX-Auroraを8VE使用する
#PBS -l elapstim_req=2:00:00       #実行時間を2時間に設定
#PBS -v OMP_NUM_THREADS=8          #1VEあたり8コアで実行（1～8）
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×8コア並列で実行
```

AOCCの場合

【逐次実行】

```
#!/bin/sh                          #シェルを指定
#PBS -q lx -b 1                    #LX 460Rz-2を1ノード使用する
#PBS -l elapstim_req=2:00:00       #実行時間を2時間に設定
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
```

【OpenMP並列実行】

```
#!/bin/sh                          #シェルを指定
#PBS -q lx -b 1                    #LX 460Rz-2を1ノード使用する
#PBS -l elapstim_req=2:00:00       #実行時間を2時間に設定
#PBS -v OMP_NUM_THREADS=128        #1ノードあたり128コアで実行（1～128）
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実行】

```
#!/bin/sh                          #シェルを指定
#PBS -q lx -b 2                    #LX 460Rz-2を2ノード使用する
#PBS -l elapstim_req=2:00:00       #実行時間を2時間に設定
#PBS -T openmpi                    #OpenMPIライブラリを使うことを指定
cd $PBS _O_WORKDIR                 #qsubを実行したディレクトリに移動
mpirun $NQSV_MPIOPTS  -np 256 ./a.out    #カレントディレクトリのa.outを256プロセスで実行
```

【MPIとOpenMP並列実行の同時利用】

```
#!/bin/sh                          #シェルを指定
#PBS -q lx -b 2                    #LX 460Rz-2を2ノード使用する
#PBS -l elapstim_req=2:00:00       #実行時間を2時間に設定
#PBS -T openmpi                    #OpenMPIライブラリを使うことを指定
#PBS -v OMP_NUM_THREADS=64         #1ノードあたり64コアで実行（1～128）
cd $PBS _O_WORKDIR                 #qsubを実行したディレクトリに移動
mpirun $NQSV_MPIOPTS --map-by ppr:2:node  -np 4 ./a.out
                                   #カレントディレクトリのa.outを4プロセス×64コア並列で実行
```

Intel Compilerの場合

【逐次実行】

```
#!/bin/sh                                  #シェルを指定
#PBS -q lx -b 1                            #LX 460Rz-2を1ノード使用する
#PBS -l elapstim_req=2:00:00               #実行時間を2時間に設定
source /opt/intel/bin/compilervars.sh intel64   #プログラム実行環境をIntelコンパイラに切替え(またはcompilervars.csh )
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
```

【OpenMP並列実行】

```
#!/bin/sh                                  #シェルを指定
#PBS -q lx -b 1                            #LX 460Rz-2を1ノード使用する
#PBS -l elapstim_req=2:00:00               #実行時間を2時間に設定
#PBS -v OMP_NUM_THREADS=128                #1ノードあたり128コアで実行（1～128）
source /opt/intel/bin/compilervars.sh intel64   #プログラム実行環境をIntelコンパイラに切替え(またはcompilervars.csh )
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実行】

```
#!/bin/sh                                  #シェルを指定
#PBS -q lx -b 2                            #LX 460Rz-2を2ノード使用する
#PBS -l elapstim_req=2:00:00               #実行時間を2時間に設定
#PBS -T intmpi                             #Intel MPIライブラリを使うことを指定
source /opt/intel/bin/compilervars.sh intel64   #プログラム実行環境をIntelコンパイラに切替え(またはcompilervars.csh )
cd $PBS _O_WORKDIR                         #qsubを実行したディレクトリに移動
mpirun $NQSV_MPIOPTS  -np 256 ./a.out      #カレントディレクトリのa.outを256プロセスで実行
```

【MPIとOpenMP並列実行の同時利用】

```
#!/bin/sh                                  #シェルを指定
#PBS -q lx -b 2                            #LX 460Rz-2を2ノード使用する
#PBS -l elapstim_req=2:00:00               #実行時間を2時間に設定
#PBS -T intmpi                             #Intel MPIライブラリを使うことを指定
#PBS -v OMP_NUM_THREADS=64                 #1ノードあたり64コアで実行（1～128）
source /opt/intel/bin/compilervars.sh intel64   #プログラム実行環境をIntelコンパイラに切替え(またはcompilervars.csh )
cd $PBS _O_WORKDIR                         #qsubを実行したディレクトリに移動
mpirun -hostfile ${PBS_NODEFILE} -ppn 2 -np 4 ./a.out
                                           #カレントディレクトリのa.outを4プロセス×64コア並列で実行
```

ジョブの実行方法と実行キュー（5/5）

サブシステムAOBA-A（SX-Aurora TSUBASA）

実行キュー名	利用可能VE数	実行時間制限 規定値／最大値	ジョブの実行形態
sxf	1	1時間／1時間	1VEジョブ 1時間無料 (VH を共用する)
sx	1	72時間／720時間	1VEジョブ (VH を共用する)
	2～256		8VE 単位で確保 (VH を共用しない)
sxmix	2～8		1VE 単位で確保 (VH を共用する)
個別設定	契約VE数	720時間	占有利用

サブシステムAOBA-B（LX 406Rz-2）

実行キュー名	利用可能ノード数	実行時間制限 規定値／最大値	ジョブの実行形態
lx	1～16	72時間／720時間	ノードを共用しない
個別設定	契約ノード数	720時間	占有利用

・バイナリを作成するのに利用したコンパイラと，投入する計算機のキューの確認が必要

→ GNUコンパイラで作成したバイナリをSXのキューに投入すると，VH（EPYC）で実行されてしまう

利用申請方法（利用者番号の取得）

センターに利用申請　https://www.ss.cc.tohoku.ac.jp/apply-for-use/ をご参照ください。

・大学・学術利用

　□ 負担金請求あり，随時申請可能

・民間企業利用（成果公開型／成果非公開型）

　□ 負担金請求あり，課題審査あり，トライアルユースあり，随時申請可能

・センターとの共同研究（大学・学術・民間企業利用対象）

　□ 負担金請求あり，共同研究割引あり，応募期間あり

各機関での課題募集

・学際大規模情報基盤共同利用・共同研究拠点公募型共同研究（JHPCN）

　□ 採択予算超過の場合に負担金請求あり，応募期間あり

・革新的ハイパフォーマンス・コンピューティング・インフラ（HPCI）

　□ 負担金請求なし（採択資源量まで利用可能），応募期間あり

利用者講習会・利用相談・高速化支援

利用者支援　https://www.ss.cc.tohoku.ac.jp/support/ をご参照ください。

・利用者講習会
- □ システムの利用法，コードの高速化・並列化，ネットワーク・セキュリティ，アプリケーション利用方法について，年間10回程度開催
- □ センター内端末機室および遠隔配信で実施

・利用相談
- □ 利用申請の方法，システムの利用方法，コンパイルエラー，ジョブの投入方法，コードの高速化など
- □ 利用相談フォームで受け付け https://www.ss.cc.tohoku.ac.jp/consultation/
- □ メールで継続的にサポート
- □ 年間約100件

・高速化支援
- □ コードを預かり，利用者，センター教職員，ベンダーが協力してコードの高速化・並列化を実施
- □ コード大規模化のサポート，JHPCN課題，HPCI課題へのステップアップを支援
- □ 1997年から継続的な取り組み
- □ 年間5〜10件程度を実施
- □ SX-ACEシステム（2015年度〜2020年度）ではベクトル高速化またはMPI並列化を30件実施
  （平均16.7倍）　（平均約2.4倍）

今後の講習会予定

No.	講習会名	開催日時	募集人数	講師	内容
1	新スーパーコンピュータ AOBAの紹介と利用説明会	11月5日(木) 13:30-15:00		サイバーサイエンスセンター 情報部情報基盤課	・新スーパーコンピュータAOBAの紹介 ・利用方法、利用負担金について
2	SX-Aurora TSUBASAへの移植に関する ハンズオンセミナー	11月20日(金) 13:30-15:30	10	NEC	・SX-ACEとの違いの紹介 ・移植ツール、移植方法について
3	はじめてのLinux	11月25日(水) 13:30-15:30	10	情報部情報基盤課	・Linuxシステムの基本的な使い方 ・エディタの使い方
4	はじめてのスパコン	11月27日(金) 13:30-15:30	10	情報部情報基盤課	・スーパーコンピュータの紹介と利用法入門
5	はじめての高速化	12月1日(火) 13:30-16:30	10	サイバーサイエンスセンター	・スーパーコンピュータの高速化について
6	並列プログラミング入門I（OpenMP）	12月9日(水) 13:30-16:30	10	サイバーサイエンスセンター	・並列プログラミングの概要 ・OpenMPによる並列プログラミングの基礎 ・利用法
7	並列プログラミング入門II（MPI）	12月11日(金) 13:30-16:30	10	サイバーサイエンスセンター	・MPIによる並列プログラミングの基礎 ・利用法

・後日以下ページのリンクに参加登録ページを公開予定

https://www.ss.cc.tohoku.ac.jp/lectures/