

国立大学法人 大阪大学
サイバーメディアセンター 御中

SX-ACEユーザー向けSQUID説明会

SX-ACEからSX-Aurora TSUBASAへの移行の流れ

2021年01月26日
日本電気株式会社
第一官公ソリューション事業部

Orchestrating a brighter world

NECは、安全・安心・公平・効率という
社会価値を創造し、
誰もが人間性を十分に発揮できる
持続可能な社会の実現を目指します。

目 次

1. はじめに

2. 移行前の準備

※ **現行システム (SX-ACE上)で準備**しておく推奨事項です。

3. プログラミング上の違い

- ① コマンド等の違い
- ② ライブラリの違い
- ③ エンディアンの違い
- ④ ローカル変数の扱い方の違い
- ⑤ 演算器の違い

4. 情報入手方法

5. 分散メモリ並列化

1. はじめに

目的

- SX-ACE (現行システム) からSX-Aurora TSUBASA(次期システム)に変わるにあたり、現行システムで動作していたアプリケーションを、次期システムへ移行するために必要な情報をまとめています。

本書の利用の仕方

SX-ACEからSX-Aurora TSUBASA に変わる際の変更点を体系的に確認されたい方

2章、3章を順番にご確認ください。

移行に際して、ご自身のアプリケーションに、どのような修正が発生するかを確認されたい方

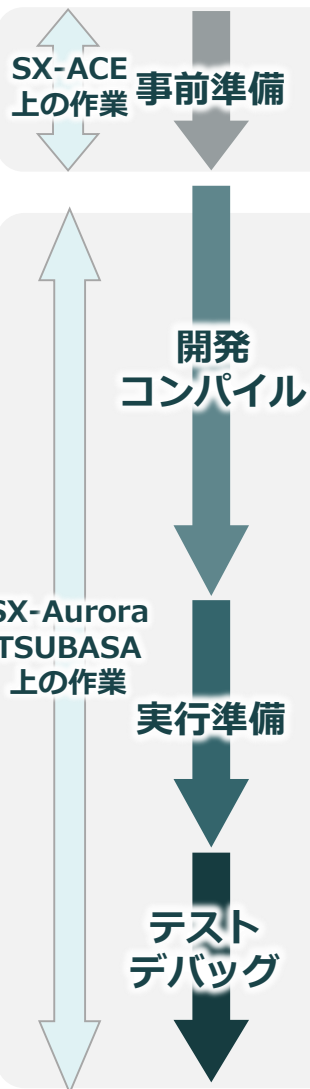
次頁の作業フローをご確認いただき、該当する項目の「本書の参照先」をご確認ください。

移行作業時のトラブルの解決方法を知りたい方

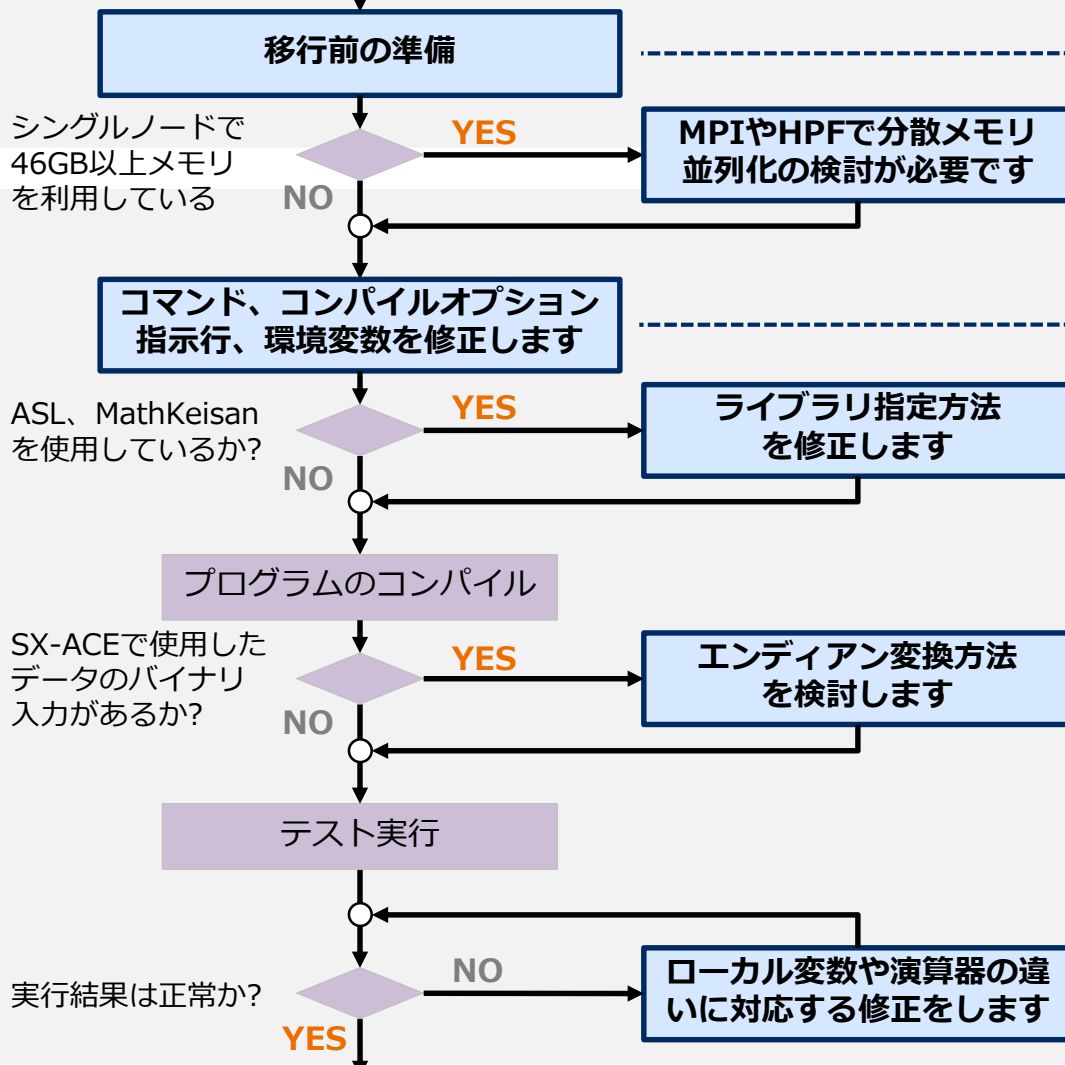
4章にて、さらなる情報の入手先をまとめています。

移行作業フロー

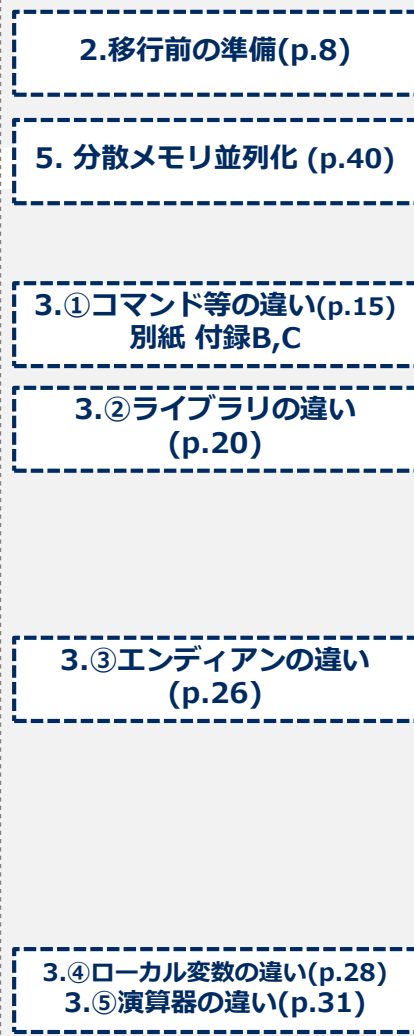
作業段階



START ○ SX-ACEの利用



本書の参照先



GOAL ○ SX-Aurora TSUBASAの利用

2. 移行前の準備

現行システム (SX-ACE上)で準備しておく
推奨事項です。

コンパイラ出力の保存

移行前にSX-ACE上で情報を採取しておくことを推奨します

移行後に気になる点があった場合に、SX-ACEとの差異や状況確認をしやすいするため

コンパイラ出力の保存

- ベクトル化、並列化情報の保存 [-pvctl fullmsg]
- 編集リストの保存 [-R5 / -Rfmtlist]
- 性能情報の出力[-ftrace] (次頁にて説明)

- Fortran90 コンパイラの実行例

```
$ sxf90 -Wf,-pvctl fullmsg -R5 -ftrace (ソースコード: f90)
```

- Fortran90 コンパイラ(MPI)の実行例

```
$ sxmpif90 -Wf,-pvctl fullmsg -R5 -ftrace (ソースコード: f90)
```

- Fortran2003 コンパイラの実行例

```
$ sxf03 -pvctl fullmsg -Rfmtlist -ftrace (ソースコード: f03)
```

- Fortran2003 コンパイラ(MPI)の実行例

```
$ sxmpif03 -pvctl fullmsg -Rfmtlist -ftrace (ソースコード: f03)
```

実行結果、性能情報の保存

実行結果、性能情報の保存

- 現行システム(SX-ACE) 上での**実行結果、実行時間を保存**してください。

次期システム(SX-Aurora TSUBASA)上で、問題が発生した際の調査に役立ちます。

- ftrace 情報を保存してください。

コンパイルオプションに-ftrace をつけた状態でアプリケーションを実行すると、性能情報(ftrace.out)がプロセス毎に生成されます。下記コマンドを実行し**テキスト化して保存**します。

- 非MPIプログラム

```
$ sxfttrace -f ftrace.out > (テキストファイル名)
```

- MPIプログラム(ランク毎)

```
$ sxfttrace -f ftrace.out.0.[rank] > (テキストファイル名)
```

- MPIプログラム(全ランク)

```
$ sxfttrace -f ftrace.out.* -all > (テキストファイル名)
```

メモリ使用量の確認

実行解析情報(PROGINF)の確認

- メモリ使用量が**46GBに収まっているか**を確認します。

46GB以上の場合、SX-Aurora TSUBASA上では**実行できない可能性があります**。

※ SX-Aurora TSUBASAのメモリ搭載量は48GBですが、メモリ使用量が増えるケースがあり余裕を見て46GBとしています。

- PROGINF 情報を確認してください。

実行時の環境変数 F_PROGINFにYESもしくはDETAILを設定しプログラムを実行すると、標準エラー出力結果に実行解析情報(PROGINF)が出力されます。

- 非MPIプログラム

```
#PBS -v F_PROGINF=YES
```

```
***** Program Information *****
Real Time (sec)      :      0.983081
User Time (sec)      :      0.982125
Sys Time (sec)       :      0.000827
Vector Time (sec)    :      0.981940
Inst. Count          :    537300463.
V. Inst. Count       :    167962313.
V. Element Count     :    42998350432.
V. Load Element Count :    17180000302.
FLOP Count           :    17179869322.
MOPS                  :    44156.994865
MFLOPS                :    17492.548629
A. V. Length         :      255.999990
V. Op. Ratio (%)      :      99.148356
Memory Size (MB)      :    256.031250
MIPS                  :    547.079509
I-Cache (sec)         :      0.000047
(以下略)
```

46GB (47104.00 MB) 以下になっていることを確認します。

ローカル変数の扱い方の違い

次期システムでは、ローカル変数の格納領域が変わります。現行システム(SX-ACE)上で情報を採取しておくに移行後の調査がしやすくなります。

- save属性の指定漏れ、初期化漏れの修正

- -P stack オプションを追加し、コンパイル/実行することで、異常終了、結果が変わるかチェックする(sxf90のみで非共有並列処理の際)

局所実体をスタックに割り付ける。

異常終了、結果が変わる場合はsave属性指定漏れ等を事前に解消しておく

- -init stack=snan heap=snanオプションを追加し、コンパイル/実行することで、異常終了、結果が変わるかチェックする

stackに割り付ける領域をNaNで初期化する。

異常終了、結果が変わる場合は初期化漏れ等を事前に解消しておく

- 注意点

上記修正で結果が変わる場合がありますが、プログラムとしては正しい記述、正しい挙動に近づきます。

結果が変わる場合、移行後の**次期システムでも結果が変わる可能性が高い**です。事前に確認、修正しておくことを強くお勧めします。

3. プログラミング上の違い

プログラミング上の違い

主な違い

		SX-ACE	SX-Aurora TSUBASA
①	コマンド等	※1	※1
②	ライブラリ	ASL+MathKeisan	NLC
③	エンディアン	ビッグエンディアン	リトルエンディアン
④	ローカル変数	bssセクションに割り付け※2	スタック領域に割り付け
⑤	演算器	※3	FMA演算器の採用 ※3

※1 コンパイラコマンド、コンパイラオプション、指示行指定、実行時環境変数
それぞれに違いがあります

※2 sxf90のみで非共有並列処理の際

※3 演算器、ベクトルパイプライン数、等が違います

① コマンド等の違い

相違点 と 対処方法

		SX-ACE	SX-Aurora TSUBASA
①	コマンド等	※1	※1

※1 コンパイラコマンド、コンパイラオプション、指示行指定、実行時環境変数
それぞれに違いがあります

コンパイラコマンド

● コンパイラコマンドの違い

言語	SX-ACE	SX-Aurora TSUBASA
Fortran	sxf90/f03 (クロスコンパイル)	nfort
C/C++	sxcc/sxc++ (クロスコンパイル)	ncc/nc++
MPI	mpisxf90,mpisxf03 mpisxcc,mpisxc++ (クロスコンパイル)	mpinfort mpincc,mpinc++

① コマンド等の違い

コンパイラオプション

● 主なコンパイラオプションの相違点

機能	SX-ACE	SX-Aurora TSUBASA
最適化レベル	-Chopt	-O3
	-Cvopt	-O2
自動並列化	-Pauto	-mparallel ※1
OpenMP並列化	-Popenmp	-fopenmp ※1
自動インライン展開	-pi (auto)	-finline-functions
自動インライン展開をしない	-Npi	-fno-inline-functions
バウンズチェック	-eR	-fbounds-check
ループアンロール	-O unroll	-floop-unroll
編集リスト出力	-L fmtlist	-report-format
プリプロセス	-Ep	-fpp
バージョン表示	-V	--version (コンパイルしない)

※1 自動並列とOpenMP並列の混在も可能

- 詳細は各コンパイラユーザズガイド内の「付録 B SX シリーズ向けコンパイラとの対応」を参照
<https://www.hpc.nec/documentation>
→「C/C++ Compiler ユーザズガイド」 「Fortran Compiler ユーザズガイド」

① コマンド等の違い

コンパイラ指示行

● 指定方法

言語	SX-ACE	SX-Aurora TSUBASA
Fortran	!CDIR	!NEC\$ 自由形式 *NEC\$ 固定形式
C/C++	#pragma cdir	#pragma _NEC

● 主なコンパイラ指示行の相違点

機能	SX-ACE	SX-Aurora TSUBASA
配列に依存関係がないことを指定	nodep	ivdep
ループの入れ替えを行う	loopchg	interchange
ループの入れ替えを行わない	noloopchg	nointerchange
ループを展開する	expand	unroll_completely
N段アンローリングする	unroll=n	unroll(n)
外側ループをn段アンローリングする	outerunroll=n	outerloop_unroll
自動ベクトル化を許可しない	novector	novector
配列をADBにバッファリングする	on_adb	なし

- SX-ACE向け指示行をSX-Aurora TSUBASA 向けに自動変換するツールを提供しています。(次頁で説明)

① コマンド等の違い

指示行変換ツール

- Fortran: nfdirconv コマンド、 C/C++: ncdirconv コマンド
 - ・指示行は 指示行変換ツールで一括変換が可能です。
 - ・念のため、変換前後を確認し、さらに、コンパイル時に編集リストを採取して、意図した変換とベクトル化となっているかを確認することをお勧めします。
- 代表的なオプション例

オプション	説明
-a	sxf90/sxf03/sxcc /sxc++のコンパイラ指示行を削除せずに、nfort/ncc /nc++のコンパイラ指示行を追加します。
-p	sxf90/sxf03/sxcc /sxc++のコンパイラ指示行に対応するnfort/ncc /nc++のコンパイラ指示行が無い場合、sxf90/sxf03/sxcc /sxc++のコンパイラ指示行を削除しません。(既定の動作)
-r	ディレクトリと同時に指定したとき、サブディレクトリを再帰的に走査します。ディレクトリのシンボリックリンクは無視します。

- 使用例

```
$ nfdirconv sample.f
```

- 詳細は各コンパイラユーザズガイド内の「付録 C コンパイラ指示行変換ツール」を参照
<https://www.hpc.nec/documentation>
→「C/C++ Compiler ユーザズガイド」 「Fortran Compiler ユーザズガイド」

① コマンド等の違い

実行時環境変数

● 主な実行時環境変数の相違点

機能	SX-ACE	SX-Aurora TSUBASA
実行解析情報の出力	F_PROGINF	VE_PROGINF
入出力バッファサイズ指定	F_SETBUF	VE_FORT_SETBUF
ファイル装置接続	F_FFn	VE_FORTn
ビッグエンディアンモード指定	F_UFMTENDIAN	VE_FORT_UFMTENDIAN
自動並列数の指定	F_RSVTASK	OMP_NUM_THREADS VE_OMP_NUM_THREADS ※1
トレースバック情報の出力	F_TRACEBACK	VE_TRACEBACK
MPI 実行性能情報の出力	MPIPROGINF	NMPI_PROGINF
MPI 通信情報の出力	MPICOMMINF	NMPI_COMMINF

※1 自動並列数とOpenMPスレッド数の指定方法は同じ

■ 詳細は各コンパイラユーザズガイド内の「付録 B.4 環境変数」を参照

<https://www.hpc.nec/documentation>

→「C/C++ Compiler ユーザズガイド」 「Fortran Compiler ユーザズガイド」

② ライブラリの違い

相違点

		SX-ACE	SX-Aurora TSUBASA
②	ライブラリ	ASL+MathKeisan	NLC

NEC Numeric Library Collection

- SX-ACEの数値計算ライブラリASL、LAPACKなどを包含するライブラリです。
- コンパイル、リンク時のオプションがSX-ACEの時と異なります。

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	FFTW3 (Version 3.x) のAPIでASLのフーリエ変換を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		連立一次方程式、固有値方程式、特異点分解
ScaLAPACK		連立一次方程式、固有値方程式、特異点分解（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（分散メモリ並列用）
SBLAS		スパース行列の基本演算
Hetero Solver		連立一次方程式（スパース行列用の直説法ソルバ）
Stencil Code Accelerator		ステンシル計算の加速

● MathKeisanでサポートしていた以下のライブラリはNLCで非サポートとなります。

- ARPACK/PARPACK
- FFT/PARFFT

② ライブラリの違い

コンパイル環境設定の実施

- プログラムのコンパイルを行う前に、コンパイル環境設定スクリプトを実行します。
- コンパイルするプログラムの分散並列の有無、および整数型のサイズによってスクリプト実行時の引数が異なります。

分散 (MPI) 並列版 ライブラリの利用	プログラムでの 既定値整数型サイズ	スクリプト指定引数
無	32bit (kind=4)	(なし)
無	64bit (kind=8)	i64
有	32bit (kind=4)	mpi

例) 64bit整数型を使用したMPI並列版ライブラリを使用する場合 (sh形式の場合)

```
$ source /opt/nec/ve/nlc/2.1.0/bin/nlcvars.sh mpi i64
```

ジョブスクリプト内での設定

- 環境設定は、プログラム実行時に必要な環境変数も設定されるため、ジョブスクリプト内でも必要です。

② ライブラリの違い

コンパイルの実施

- 32bit整数型、64bit整数型ライブラリを利用するかでコンパイラオプションが異なります。
(1) 32bit整数版ライブラリを使用する場合

ライブラリ名		コンパイラ オプション	リンクオプション
ASLネイティブ インタフェース	逐次	なし	-lasl_sequential
	OpenMP並列	-fopenmp	-lasl_openmp -fopenmp
ASL統合 インタフェース	逐次	なし	-lasl_sequential
	OpenMP並列	-openmp	-lasl_openmp -fopenmp
	MPI並列	なし	-lasl_mpi_sequential
	ハイブリッド並列	-openmp	-lasl_mpi_openmp -fopenmp
FFTW3 インタフェース	逐次	なし	-laslfftw3 -lasl_sequential
	OpenMP並列	-openmp	-laslfftw3 -lasl_openmp -fopenmp
	MPI並列	なし	-laslfftw3_mpi -lasl_mpi_sequential
	ハイブリッド並列	-fopenmp	-laslfftw3_mpi -lasl_mpi_openmp -fopenmp

- 上記以外のライブラリに関するオプションは、NLCユーザズガイド内の「コンパイルとリンク」を参照
[Fortran用] https://www.hpc.nec/documents/sdk/SDK_NLC/UsersGuide/main/ja/f_linking.html
[C言語用] https://www.hpc.nec/documents/sdk/SDK_NLC/UsersGuide/main/ja/c_linking.html

② ライブラリの違い

コンパイルの実施

(2) 64bit整数版ライブラリを使用する場合

ライブラリ名		コンパイラオプション	リンクオプション
ASL ネイティブ インタフェース	逐次	-fdefault-integer=8	-lasl_sequential_i64
	OpenMP並列	-fdefault-integer=8 -fopenmp	-lasl_openmp_i64 -fopenmp
ASL統合 インタフェース	逐次	-fdefault-integer=8	-lasl_sequential_i64
	OpenMP並列	-fdefault-integer=8 -fopenmp	-lasl_openmp_i64 -fopenmp
	MPI並列	-fdefault-integer=8 -fdefault-real=8	-fdefault-integer=8 -fdefault-real=8 -lasl_mpi_sequential_i64f
	ハイブリッド 並列	-fdefault-integer=8 -fdefault-real=8 -fopenmp	-fdefault-integer=8 -fdefault-real=8 -lasl_mpi_openmp_i64f -fopenmp
FFTW3 インタフェース	逐次	-fdefault-integer=8	-laslfftw3_i64 -lasl_sequential_i64
	OpenMP並列	-fdefault-integer=8 -fopenmp	-laslfftw3_i64 -lasl_openmp_i64 -fopenmp
	MPI並列	-fdefault-integer=8 -fdefault-real=8	-fdefault-integer=8 -fdefault-real=8 -laslfftw3_mpi_i64f - lasl_mpi_sequential_i64f
	ハイブリッド 並列	-fdefault-integer=8 -fdefault-real=8 -fopenmp	-fdefault-integer=8 -fdefault-real=8 -laslfftw3_mpi_i64f -lasl_mpi_openmp_i64f -fopenmp

② ライブラリの違い

FFTW3インタフェース利用時の留意事項

- 何点か留意事項(配列の次元順、Cの複素数型 等)があるので利用前に参照(下記)ください。
- マニュアルページでサンプルコードの表示とダウンロードが可能です。
- **ヘッダファイルの変更が必要**です。

FFTW3インタフェース (Fortran用)

概要

FFTW3インタフェースは、FFTW version 3.xとほぼ互換のインタフェースをもつライブラリです (一部のルーチンおよび定数は未対応)。FFTWのインクルードファイルを差し替えるだけで、FFTWを使用しているユーザプログラムから、Vector Engine向けに高度にチューニングされたASLのフーリエ変換ルーチンを使用することができます。オリジナルのFFTWは、[FFTW公式サイト](#)[1]で公開されているオープンソースのフーリエ変換ライブラリです。詳細は、[FFTWのドキュメント](#)[2]を参照してください。

FFTW3インタフェースの使用方法

FFTW3インタフェースのインクルードファイルは、オリジナルのFFTWで提供されているものとファイル名が異なります。FFTW3インタフェースを使用する場合は、以下のようにインクルードファイル名を変更してください。

Fortran 2003プログラムの場合

```
include "fftw3.f03"  ⇒  include "aslfftw3.f03"
```

Fortran 77プログラムの場合

```
include "fftw3.f"    ⇒  include "aslfftw3.f"
```

コンパイルとリンクには、FFTW3のコンパイル方法およびリンク方法を記載しています。

include "fftw3.f03" →	include "aslfftw3.f03"
include "fftw3.f" →	include "aslfftw3.f"
#include <fftw3.h> →	#include <aslfftw3.h>

- NLCのFFTW3インタフェースは**スレッドセーフではない**ため、共有メモリ並列と併用する際は注意が必要です。

■ 詳細は、NLCユーザズガイド内の「FFTW3インタフェース」を参照

[Fortran用] https://www.hpc.nec/documents/sdk/SDK_NLC/UsersGuide/fftwint/f/ja/index.html

[C言語用] https://www.hpc.nec/documents/sdk/SDK_NLC/UsersGuide/fftwint/c/ja/index.html

ASL/ASLSTAT移行のお願い

ASL統合インタフェースへの移行のお願い

- SX-Aurora TSUBASAで性能を得られる新しいASL統合インタフェースを開発致しました。ご不便をおかけしますが、SX-ACEでASLおよびASLSTATの以下の機能をご利用の場合は、SX-Aurora TSUBASAではASL統合インタフェースのルーチン／関数への置換えをお願い致します。

機能	廃止状況
フーリエ変換	廃止予定
	廃止済
COSINE/SINE変換	廃止済
乱数生成	廃止済
ソート	廃止検討中

- **廃止予定**： ASL統合インタフェースの方が高速なため非推奨です。廃止時期は未定です。
- **廃止済**： ASL統合インタフェースでのみ提供しております。
- SX-Aurora TSUBASA版からSX-ACE以前のASLおよびASLSTATの機能を統廃合し、"ASLネイティブインタフェース"と名称を変更しております。
- ASLネイティブインタフェースの機能の廃止状況の詳細、および、具体的な移行方法については、別途、資料をご用意しておりますので、そちらをご参照ください。

③ エンディアンの違い

相違点 と 対処方法

		SX-ACE	SX-Aurora TSUBASA
③	エンディアン	ビッグエンディアン	リトルエンディアン

- Fortranは実行時環境変数で対処可能、Cでは環境変数はない(バイトスワップが必要)
- プリ/ポストツールとの連携に注意する
- 実行時環境変数 **VE_FORT_UFMTENDIAN** (Fortran)でビッグエンディアンを扱えます
使い方の一例

引数	説明
ALL	export VE_FORT_UFMTENDIAN=ALL 全ての装置番号をビッグエンディアンとする
番号,番号	export VE_FORT_UFMTENDIAN=10,11 装置番号10番と11番をビッグエンディアンとする
番号-番号 (連続の範囲)	export VE_FORT_UFMTENDIAN=10-12 装置番号10番から12番をビッグエンディアンとする
big:番号 little:番号 と ; の除外	export VE_FORT_UFMTENDIAN=big;little:10-12 装置番号10番から12番 以外 をビッグエンディアンとする (;前の指定から、除外する装置番号を後ろに指定する)

SX-ACE で F_UFMTENDIAN (リトルエンディアン、Fortran) を指定していたものは設定不要となります

③ エンディアンの違い

■ バイトスワップについて

- **Cでは環境変数がないため、バイトスワップが必要**となります。
- 8byte データをバイトスワップするサンプルスクリプトを記載します。
- 実際には、バイナリデータのI/Oにて、4byte/8byte など出力形式はアプリケーションにより異なります。アプリケーション毎のファイル仕様に従いバイトスワップが必要です。

endian_b2l.pl

```
#!/usr/bin/perl

rename $ARGV[0],Big_.$ARGV[0];
open $ifh, , Big_.$ARGV[0];
open $ofh, '>', $ARGV[0];

print "[",$ARGV[0],"] START¥n";
while( read($ifh, $data, 8)){
    @wdata = unpack('N2', $data);
    @wdata = ($wdata[1], $wdata[0]);
    $data = pack('V2', @wdata);
    print $ofh $data;
}
print "[",$ARGV[0],"] END¥n";

close($ifh);
close($ofh);
```

● 使用例

```
$ ./endian_b2l_v01.pl (変換対象ファイル)
```

④ ローカル変数の扱い方の違い

相違点

		SX-ACE	SX-Aurora TSUBASA
④	ローカル変数	bssセクションに割り付け※2	スタック領域に割り付け

※2 sx90のみで非共有並列処理の際

● bssセクション

スタティックな領域。手続き実行でセットされた**値が以降も保持**される。
初期値なしスタティック領域であるが、
実行に先立ち **値0(論理型はFALSE)**に初期化される。

ローカル(局所)変数であっても手続き終了後に、bss上で値が残っている。
(save属性指定漏れでも、なんとなってしまう場合がある)

● スタック領域

save属性を指定しないローカル(局所)変数は手続きが終了したら**不定**になる。

save属性漏れ、初期化漏れのローカル(局所)変数がある場合、
移行後に結果相違、実行エラー等が発生する可能性がある

④ ローカル変数の扱い方の違い

■ 事前準備の勧め ※ **現行システム(SX-ACE)上で行っておく**こと (再掲)

● save属性の指定漏れ、初期化漏れの修正

- -P stack オプションを追加し、コンパイル/実行することで、異常終了、結果が変わるかチェックする (sxf90のみで非共有並列処理の際)

局所実体をスタックに割り付ける。

異常終了、結果が変わる場合はsave属性指定漏れ等を事前に解消しておく

- -init stack=snan heap=snan オプションを追加し、コンパイル/実行することで、異常終了、結果が変わるかチェックする

stackに割り付ける領域をNaNで初期化する。

異常終了、結果が変わる場合は初期化漏れ等を事前に解消しておく

● 注意点

上記修正で結果が変わる場合がありますが、
プログラムとしては正しい記述、正しい挙動に近づきます。

結果が変わる場合、移行後の**次期システムでも結果が変わる可能性が高い**です。
事前に確認、修正しておくことを強くお勧めします。

④ ローカル変数の扱い方の違い

移行後の対処

- 次期システム(SX-Aurora TSUBASA)で対応する。
- 実行エラー、結果が変わる場合はソース修正が必要となります。
 - ・ソースコードからsave属性漏れを探し、save属性を指定する
 - ・ソースコードから初期化漏れを探し、明示的な初期化を行う
- コンパイラオプションで対処する(次善の策) (性能が低下する場合があります)

実行時の例外検出方法
倍精度の場合
コンパイラオプション
-minit-stack=snan
実行時環境変数
VE_INIT_HEAP=SNAN

オプション	説明
-bss	局所変数を.bssセクションに割り付ける。
-save	各プログラム単位において(RECURSIVEであるものは例外とする)、SAVE文がすべての局所変数に指定されているものとする
-minit-stack=zero	実行時にスタックに割り付ける領域を0(ゼロ)で初期化する

-bss と-save オプションの違い

SAVE属性をもつ変数の値は、前回当該ルーチンが呼び出され、リターンしたときの値が、本呼び出し時の最初の値となりますが、-bssの場合はそれが保証されません。

- 実行時環境変数で対処する(次善の策)

オプション	説明
VE_INIT_HEAP=ZERO	実行時にヒープに割り付ける領域を0(ゼロ)で初期化する

0初期化をする場合はコンパイラオプション-minit-stack=zeroと併用することをお勧めします

2. 移行前準備が役立つ場合がある

⑤ 演算器の違い

相違点

		SX-ACE	SX-Aurora TSUBASA
⑤	演算器	※3	FMA演算器の採用 ※3

※3 演算器、ベクトルパイプライン数、等が違います

● 演算器構成の違い (FMA : 積和演算)

- SX-Aurora TSUBASAではFMA演算器を採用
SX-ACEとは丸め誤差レベルで演算結果が変わる可能性がある
- 丸め誤差が気になる場合は コンパイラオプション `-mno-vector-fma` (ベクトル積和演算命令の使用を許可しない) でFMA演算を抑止することが可能

● 逆数近似処理のサポート

- 除算/SQRT演算器をHW実装している。更に高速に処理するために逆数近似処理をサポートしている

– 除算

実行速度は上から下の順で速くなる

説明	コンパイラオプション	誤差
除算器を利用	<code>-mvector-floating-divide-instruction</code>	なし
逆数近似を利用 (既定値)	<code>-mno-vector-floating-divide-instruction</code>	なし
高速版逆数近似を利用	<code>-mvector-low-precise-divide-function</code>	あり(仮数部に最大 1 ビット)

⑤ 演算器の違い

相違点

● 逆数近似処理のサポート (続き)

– SQRT

実行速度は上から下の順で速くなる

説明	コンパイラオプション	誤差
SQRT演算器を利用	-mvector-sqrt-instruction	なし
逆数近似を利用 (既定値)	-mno-vector-sqrt-instruction	あり

● ベクトル総和演算の計算結果について

- 今までのSXシリーズと同様、ベクトル総和演算の計算結果に差分が生じる可能性あり
(加算順序の違いが原因)
- ベクトル総和演算を使用しない場合はコンパイラオプション -mno-vector-reduction で指定可能
 - 本オプションを使用するとベクトル化不可となるので、総和处理のコストが重い場合は実行時間が増大する

4. 情報入手方法

SX-Aurora TSUBASA に関する情報

NEC Aurora Forum

URL : <https://www.hpc.nec/>

- 各種ドキュメント、コミュニティ



NEC Aurora Forum



Menu



JOIN TODAY
NEC SX-Aurora TSUBASA

SX-Aurora TSUBASA に関する情報

各種公開ドキュメント

URL : <https://www.hpc.nec/documentation>

● SDK (コンパイラ情報等)

- ・コンパイラオプション、ライブラリ、性能分析ツール等のマニュアル

● NLC (NEC Numeric Library)

● NEC MPI

Table 3: NEC SDK

Document Name	Revision	Target Software	Updated	Old Revision
C/C++ Compiler ユーザーズガイド	21	-	Dec 25th, 2020	-
Fortran Compiler ユーザーズガイド	21	-	Dec 25th, 2020	-
PROGINF/FTRACE ユーザーズガイド	6	-	Oct 30th, 2020	-
NEC Ftrace Viewer ユーザーズガイド	2	-	Jan 31st, 2020	-
NEC Parallel Debugger ユーザーズガイド	3	-	Jun 5th, 2019	-
NLC (NEC Numeric Library Collection) ユーザーズガイド	2.2.0	-	Dec 25th, 2020	✓
Vector Engine Assembly Language Reference Manual	1.3	-	Oct 9th, 2019	-
Instruction Set Architecture Guide	1.1	-	Mar 12th, 2019	-
System V Application Binary Interface VE Architecture Processor Supplement	2.1	-	Feb 7th, 2019	-
ELF Handling For Thread-Local Storage VE Architecture Processor Supplement	1.2	-	Feb 7th, 2019	-
NLC (NEC Numeric Library Collection) リリースノート	-	-	Dec 25th, 2020	-

Table 4: NEC MPI

Document Name	Revision	Target Software	Updated	Old Revision
NEC MPI ユーザーズガイド	15	NEC MPI v2.13.0	Dec 25th, 2020	✓

コンパイラオプションの対応表 (部分)

- Fortran Compiler ユーザーズガイド 付録B
(C/C++ Compiler ユーザーズガイド にも同様ページあり)

付録 B SX シリーズ向けコンパイラとの対応

付録 B SX シリーズ向けコンパイラとの対応

本節では SX シリーズ向けのコンパイラと Vector Engine 向けコンパイラの主なコンパイラオプション、指示行環境変数の対応について説明します。

B.2 Fortran90/SXコンパイラオプション

Fortran90/SX コンパイラオプションと Vector Engine 向けコンパイラのコンパイラオプションの対応表を示します。Vector Engine コンパイラ列の「なし」は対応するコンパイラオプションがないことを表します。

B.2.1 f90/sxf90コマンド基本オプション

Fortran90/SXコンパイラ	Vector Engineコンパイラ
-Chopt	-O3
-Cvopt	-O2
-Csopt	-O2 -mno-vector
-Cvsafe	-O1
-Cssafe	-O1 -mno-vector
-Cdebug	-O0 -g
-c	-c
-Nc	なし

SX-ACE からの移行に関する情報

環境変数の対応表 (部分)

● Fortran Compiler ユーザーズガイド 付録B

B.4 環境変数

SX シリーズ向けコンパイラの主要な実行時に参照される環境変数とほぼ同等の機能をもつ Vector Engine 向けコンパイラの環境変数を以下に示します。

SXシリーズ向けコンパイラ	Vector Engineコンパイラ
F_PROGINF	VE_PROGINF
F_TRACEBACK	VE_TRACEBACK
F_EXPRCW	VE_FORT_EXPRCW
F_FMTBUF	VE_FORT_FMTBUF

その他ライブラリ (部分)

● Fortran Compiler ユーザーズガイド 付録B

B.5 その他のライブラリ

SX シリーズ向けコンパイラのその他のライブラリの手続とほぼ同等の機能をもつ Vector Engine 向けコンパイラの手続を以下に示します。USE 文の代わりに-useの使用も可能です。

SXシリーズ向けコンパイラ	Vector Engineコンパイラ
CALL ABORT()	USE F90_UNIX CALL ABORT()
RESULT = ACCESS(NAME,MODE)	USE F90_UNIX_FILE CALL ACCESS(NAME,AMODE,RESULT) (注) MODE(文字)がAMODE(整数)に変更 されています。AMODE(整数)の詳細は「9.3.5 F90_UNIX_FILE」の パラメタを参照してください。

コンパイラ指示行変換ツールによる対応表 (部分)

- Fortran Compiler ユーザーズガイド 付録C
(C/C++ Compiler ユーザーズガイド にも同様ページあり)

付録 C コンパイラ指示行変換ツール

付録 C コンパイラ指示行変換ツール

本節では `sxf90/sxf03/sxcc/sxc++` のコンパイラ指示行を `nfort/ncc/nc++` のコンパイラ指示行に変換するツールについて説明します。

C.1 nfdirconv

コマンド名

nfdirconv

書式

nfdirconv [オプション...] [ファイル | ディレクトリ]...

C.3 コンパイラ指示行

sxf90/sxf03/sxcc/sxc++ のコンパイラ指示行と変換後のコンパイラ指示行を以下に示します。
「変換後」列の (Remained) は将来実装予定のコンパイラ指示行、(Removed/Obsolescent) はサポートしないコンパイラ指示行を表します。

SXシリーズ向けコンパイラ	変換後
<code>alloc_on_vreg(identifier, n)</code>	<code>vreg(identifier)</code>
<code>altcode</code>	<code>dependency_test</code> <code>loop_count_test</code> <code>shortloop_reduction</code>
<code>altcode=dep</code>	<code>dependency_test</code>
<code>altcode=loopcnt</code>	<code>loop_count_test</code>

NEC Numeric Library Collection マニュアル

●コンパイルとリンク(言語別にある)

NEC Numeric Library Collection 2.1.0 ユーザーズガイド

[\[English \]](#) [Japanese](#)

TOP

Fortran用

ASL

- ネイティブインタフェース
- 統合インタフェース
- FFTW3インタフェース

BLAS

LAPACK

ScaLAPACKおよびBLACS

SBLAS

HeteroSolver

Stencil Code Accelerator

コンパイルとリンク

C言語用

著作権情報および商標

再配布ポリシー

更新履歴

NEC Numeric Library Collection ユーザーズガイド (Fortran用)

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を用いることによって、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。このページはFortranプログラムから利用するユーザを想定しています。
[\(C言語用のページへ\)](#)

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(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

ページの先頭へ

39

© NEC Corporation 2020

Orchestrating a brighter world

NEC

5. 分散メモリ並列化

分散メモリ並列化手法

SX-ACE及びSX-Aurora TSUBASAでは、**MPI**と**HPF**が分散メモリ並列化手法として用意されています。

MPI(Message Passing Interface)とは？

移植性や**対応環境の多さに優れる**が、
並列化に伴う**開発者負担が大きい**

- 分散メモリ並列処理における、複数プロセス間でデータをやり取りするために用いるメッセージ通信操作の仕様標準
- FORTRAN, C, C++から呼び出すサブプログラムのライブラリ
- ポータビリティに優れている
 - ・標準化されたライブラリにより、様々なMPI実装環境で同じソースをコンパイル・実行できる
- 開発者の負担が比較的大きい
 - ・プログラムを分析して、データ・処理を分割し、通信の内容とタイミングをユーザが自身で記述する

HPF(High Performance Fortran)とは？

並列化に伴う**開発者負担が小さい**反面、**対応環境の制約が多い**

- Fortran95の分散メモリ型並列計算機向け拡張仕様で以下の特徴を有する
 - ・国際的な標準仕様(並列処理を主要ベンダ、大学、研究機関で共同で仕様策定)
 - ・記述が平易(Fortran + コメント形式の指示文)
 - ・上級ユーザは細かな制御も可能(通信の制御、部分的にMPIを利用したチューニング等)

MPIとHPF の役割分担

MPIとHPFで、システム側が自動制御するもの、開発者が手動制御するものの関係は以下の通りです。

分散並列プログラム開発における、主要な処理要素

- ① 複数CPU,コアへの**データの分割配置(マッピング)**
- ② 複数CPU,コアへの**並列化・処理の分担**
- ③ 複数CPU,コア間の**通信処理の挿入**

処理要素	MPI	HPF
①データの分割配置	手動	手動 ※
②並列化・処理の分担	手動	自動
③通信処理の挿入	手動	自動

※SX-Aurora TSUBASA環境(NEC HPF)では、データの分割配置を一部自動化するデータ分散指定オプションが利用可能です。

MPIとHPFの比較

コード例

Fortran

```
parameter(n=100)
real*4 x(n),y(n)
do i=1,n
  x(i) = 1.e0 / float(i)
enddo
y = 0.e0
do i=2,n-1
  y(i) = (x(i-1)+2*x(i)+x(i+1))*0.25
enddo
dot=0.e0
do i=1,n
  dot = dot+ y(i)*y(i)
enddo
write(6,*)'dot =', dot
end
```

HPF

```
parameter(n=100)
real*4 x(n),y(n)
!hpf$ distribute (block) :: x,y ! データマッピング
do i=1,n
  x(i) = 1.e0 / float(i)
enddo
y = 0.e0
do i=2,n-1
  y(i) = (x(i-1)+2*x(i)+x(i+1))*0.25
enddo
dot=0.e0
do i=1,n
  dot = dot+ y(i)*y(i)
enddo
write(6,*)'dot =', dot
end
```

行数比較

Fortran **15**

HPF **16**

MPI **42**

MPI (4並列)

```
include 'mpif.h'
parameter(n=25)
real*4 x(0:n+1),y(n),dotr ! 宣言の変更, 袖の付加
integer ist(mpi_status_size)
call mpi_init(ierr) ! 初期化
call mpi_comm_size(mpi_comm_world,np,ierr)
call mpi_comm_rank(mpi_comm_world,id,ierr)
do i=1,n
  x(i) = 1.e0 / float(i+50*id)
enddo
y = 0.e0
lb = 1
ub = n ! データ転送
isnd1=id+1
ircv1=id-1
isnd2=id-1
ircv2=id+1
if(id.eq.0)then
  ircv1=mpi_proc_null
  isnd2=mpi_proc_null
  lb=2
else if(id.eq.np-1)then
  isnd1=mpi_proc_null
  ircv2=mpi_proc_null
  ub=n-1
endif
call mpi_sendrecv(x(n),1,mpi_real,isnd1,0,
& x(1),1,mpi_real,ircv1,0, mpi_comm_world,ist,ierr)
call mpi_sendrecv(x(1),1,mpi_real,isnd2,1,
& x(n+1),1,mpi_real,ircv2,1, mpi_comm_world,ist,ierr)
do i=lb,ub
  y(i) = (x(i-1)+2*x(i)+x(i+1))*0.25
enddo
dot=0.e0
do i=1,n
  dot = dot+ y(i)*y(i)
enddo
call mpi_reduce(dot,dotr,1,mpi_real,mpi_sum,0, ! データ転送
& mpi_comm_world,ierr)
if(np.eq.0)write(6,*)'dot =', dotr
call mpi_finalize(ierr) ! 終了処理
end
```

分散メモリ並列化 関連情報

分散メモリ並列化に関する過去の講習会情報

※ 講習会資料の閲覧ができます

- 並列プログラミング入門(MPI)

http://www.hpc.cmc.osaka-u.ac.jp/lec_ws/20200703/

SX-Aurora TSUBASAのMPI利用(NEC MPI)

- NEC MPI ユーザーズガイド

<https://www.hpc.nec/documentation>

→ 「NEC MPI ユーザーズガイド」

SX-Aurora TSUBASAのHPF利用(NEC HPF)

※ HPFの概要(スタートガイド)、利用方法(ユーザーズガイド)が閲覧できます。

- NEC High Performance Fortran (HPF) Language Processor Released

<https://www.hpc.nec/forums/topic?id=Lb2cmG#L6yF51>

 **Orchestrating** a brighter world

NEC