ISSN 2436-0066

TOHOKU
UNIVERSITY

東北大学
サイバーサイエンスセンター

大規模科学計算システム広報

SENAC

Vol.53 No.4 2020−10

Cyberscience
Center

Supercomputing System
Tohoku University
www.ss.cc.tohoku.ac.jp

大規模科学計算システム関連案内

＜大規模科学計算システム関連業務は、サイバーサイエンスセンター本館内の情報部情報基盤課が担当しています。＞

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

階	係・室名	電話番号(内線)* e-mail	主なサービス内容	サービス時間 平 日
一階	利用相談室	022-795-6153(6153) sodan@cc.tohoku.ac.jp 相談員不在時 022-795-3406(3406)	計算機利用全般に関する相談 大判プリンタ、利用者端末等の利用	8:30～17:15 8:30～21:00
	利用者談話室	(3444)	各センター広報の閲覧 自販機	8:30～21:00
	展 示 室*（分散コンピュータ博物館）*	*見学希望の方は共同利用支援係までご連絡ください。	歴代の大型計算機等の展示	9:00～17:00
三階	総務係	022-795-3407(3407) cc-som@grp.tohoku.ac.jp	総務に関すること	8:30～17:15
	会計係	022-795-3405(3405) cc-kaikei@grp.tohoku.ac.jp	会計に関すること、負担金の請求に関すること	8:30～17:15
	共同利用支援係 （受 付）	022-795-3406(3406) 022-795-6251(6251) uketuke@cc.tohoku.ac.jp	利用手続き、利用相談、講習会、ライブラリ、見学、アプリケーションに関すること	8:30～17:15
	共同研究支援係	022-795-6252(6252) rs-sec@cc.tohoku.ac.jp	共同研究、計算機システムに関すること	8:30～17:15
	ネットワーク係	022-795-6253(6253) i-network@grp.tohoku.ac.jp	ネットワークに関すること	8:30～17:15
	情報セキュリティ係	022-795-3410(3410) i-security@grp.tohoku.ac.jp	情報セキュリティに関すること	8:30～17:15
四階	研究開発部	022-795-6095(6095)		
五階	端末機室	(3445)	PC 端末機(X 端末)	8:30～17:15

＊（　）内は東北大学内のみの内線電話番号です。青葉山・川内地区以外からは頭に 92 を加えます。

本誌の名前「SENAC」の由来

昭和 33 年に東北地区の最初の電子計算機として、東北大学電気通信研究所において完成されたパラメトロン式計算機の名前で SENAC-1(SENdai Automatic Computer-1)からとって命名された。

［大規模科学計算システム］

SX-Aurora TSUBASA のハードウェア

星 宗王　江副 健司　上山根 慎
日本電気株式会社

1. はじめに

東北大学サイバーサイエンスセンターで本年 10 月より新しいスーパーコンピュータ AOBA の運用が開始されます。AOBA はベクトル型システムであるサブシステム AOBA-A と、x86 システムであるサブシステム AOBA-B から構成されています。本稿では、AOBA-A を構成する SX-Aurora TSUBASA のハードウェアを紹介します。

近年、シミュレーション解析のニーズをはじめとする科学技術計算需要はますます増大しており、一方で大規模化するシステムに対して省電力化、省スペース化の要求もますます高まっています。この要請に応えるために、NEC は従来のベクトル型コンピュータを PCI Express カードのフォームファクタにパッケージし、ユーザが x86 ベースのサーバからシームレスにベクトル型コンピュータを活用できるシステムアーキテクチャを採用した、新しいスーパーコンピュータ SX-Aurora TSUBASA シリーズを 2018 年 2 月に出荷開始しました。

このたび東北大学サイバーサイエンスセンターに導入いただいたシステムは、前機種よりもメモリ性能をさらに強化し、2020 年から提供を開始した第二世代製品になります。

本稿では、AOBA-A を構成する SX-Aurora TSUBASA システムのアーキテクチャ、システム概要、ハードウェア構成、テクノロジについて紹介します。

2. Aurora アーキテクチャ

SX-Aurora TSUBASA は、NEC のベクトル型スーパーコンピュータ SX シリーズの流れを汲む製品として 2018 年から出荷開始した新しいタイプのスーパーコンピュータです。Vector Engine と呼ばれる PCI Express カードにベクトルプロセッサと主記憶を搭載し、これを Vector Host と呼ばれる標準的な x86 サーバに接続することによってシステムが構成されます。

Vector Engine に搭載されるベクトルプロセッサは、従来の SX シリーズのベクトルプロセッサ構成を踏襲しています。通常、サーバに接続される GPU や FPGA のような PCI Express カード型のアクセラレータは、ホスト側で実行されるアプリケーションの一部分を実行し、全体の処理時間の短縮を図ります。一方 Vector Engine は、コンパイルされたアプリケーション実行ファイルを丸ごとカード上で実行する、Aurora アーキテクチャを採用しています。図 1 に Aurora アーキテクチャにおける実行モデルを示します。

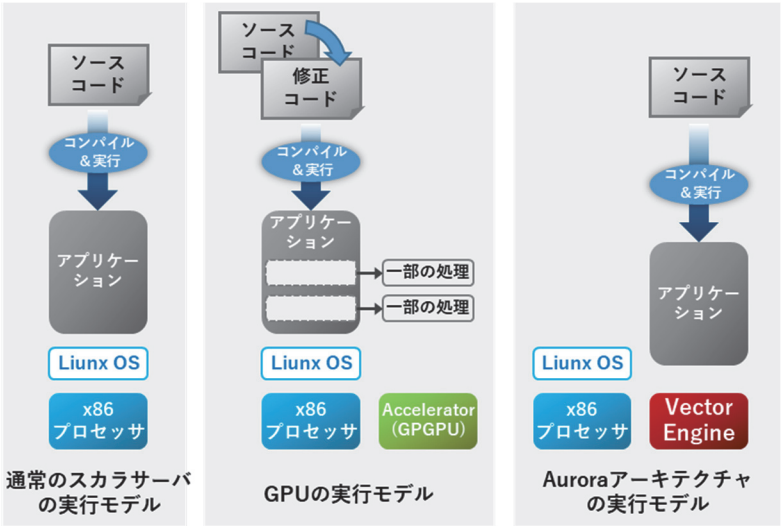


図 1. Aurora アーキテクチャの実行モデル

Aurora アーキテクチャでは、原則としてアクセラレータ向けのプログラム修正を行うことなく、SX-Aurora TSUBASA 向けコンパイラを用いてコンパイルしたプログラムをそのまま実行することができます。(さらに高い実行性能を得るためには、Vector Engine 向けのコードチューニングを実施することが望ましいです。)

この実行モデルにはいくつかのメリットがあります。第一に、ユーザにとって使い慣れた一般的な Linux OS 環境から、SX シリーズの高性能なベクトルプロセッサを利用できます。SX シリーズでは従来、専用の OS 環境を提供していましたが、SX-Aurora TSUBASA は標準的な x86 サーバと Linux OS 環境から演算処理だけにベクトルプロセッサを利用することができます。第二に、プログラム全体をVector Engineカード上で実行することにより、PCI Express バス上の頻繁なデータ移送を避け、性能ボトルネックを解消することができます。

3. システム概要

AOBA-A は、Vector Engine カードを 8 枚搭載した Vector Host(SX-Aurora TSUBASA B401-8)計 72 台で構成され、各 Vector Host は InfiniBand Network によって接続されます。

システムの主要な諸元を表 1 に示します。

表 1. AOBA-A の諸元

Vector Engine	モデル名	Type 20B
	CPU コア数	8
	理論ベクトル演算性能	2.45 TFlops(DP)
	メモリ容量	48 GB
	メモリ帯域	1.53 TB/s
	インタフェース	PCI Express Gen3 x16
Vector Host	CPU モデル名	AMD EPYC 7402P
	CPU コア数	24
	メモリ容量	256 GB
	理論演算性能	1.075TFlops(DP)
	OS	CentOS
	搭載 Vector Engine 数	8
	ネットワーク I/F	InfiniBand HDR x2
システム	Vector Host 数	72
	Vector Engine 数	576
	CPU コア数	6,336
	総理論演算性能	1,488.6 TFlops
	総メモリ容量	45TB
	総メモリ帯域	895.68TB/s

4. ハードウェア構成

4.1 Vector Engine Type20B

AOBA-A は最新の Vector Engine Type 20B を合計 576 枚搭載しています。Vector Engine は前述したように PCI Express カードのフォームファクタにベクトルプロセッサと主記憶を搭載した SX-Aurora TSUBASA の心臓部です。図 2 に Vector Engine カードの外形を、表 2 にカードの実装仕様を、図 3 にブロック図を示します。

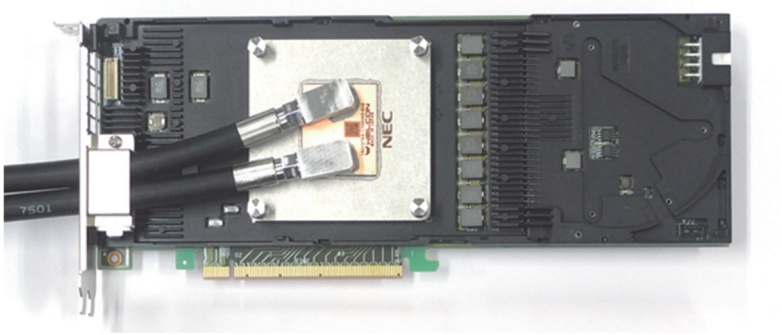


図 2. Vector Engine カード（水冷タイプ）

表 2. Vector Engine カード実装仕様（水冷ホースを除く）

Board Length	10.5 inch
Board Height	4.376 inch（PCI Express CEM3.0 maximum）
Z-Height of Component side	最大 1.370 inch（PCI Express CEM3.0×2slot 分）
Z-Height of Solder side	0.105 inch
Power Connector type	8-pin EPS 12V
Power Connector Location	EAST

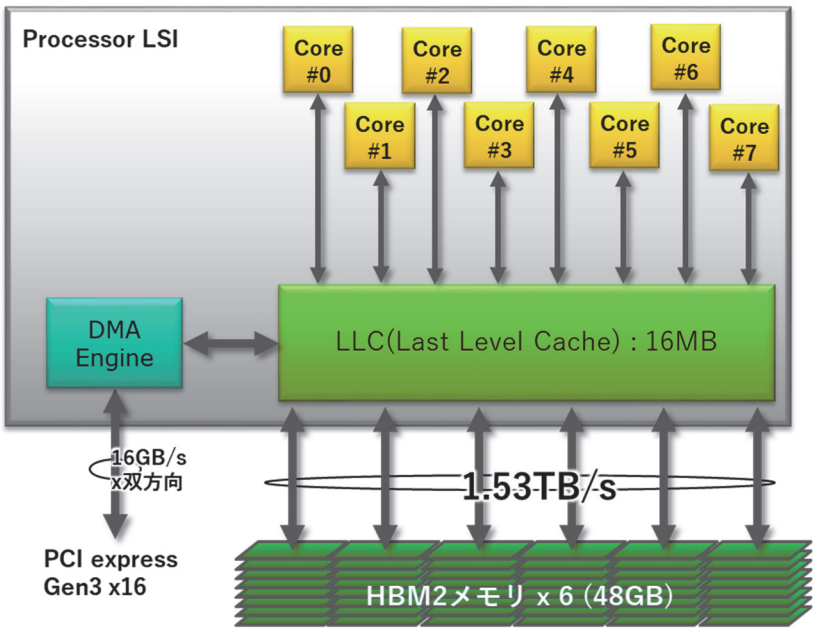


図 3. Vector Engine ブロック図

Vector Engine は 8 個の CPU コアとコア間で共有される LLC(Last Level Cache)、DMA エンジン、PCI Express インタフェース、6 個の HBM2 メモリで構成されます。

各 CPU コアは、基本的には従来の SX アーキテクチャを継承しつつ、マイクロアーキテクチャレベルで多くの改良を加えた新しいベクトルプロセッサコアです。図 4 に CPU コアの構成を示します。

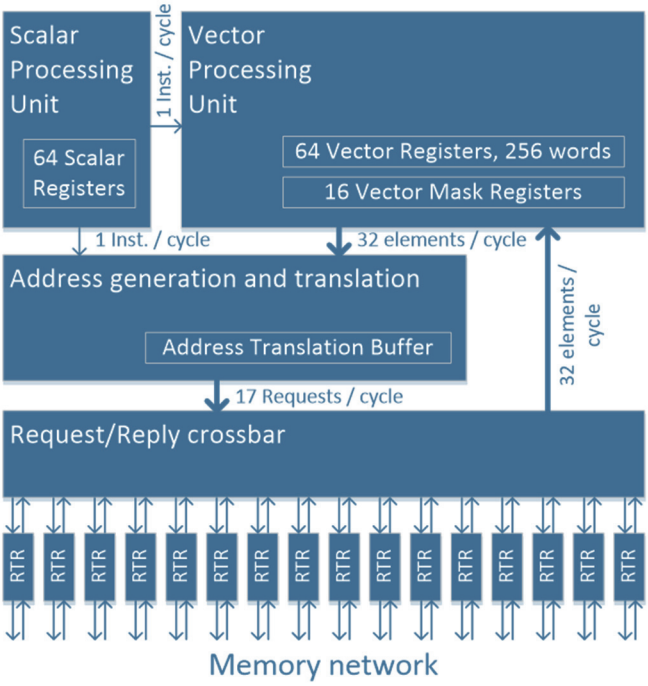


図 4. CPU コア ブロック図

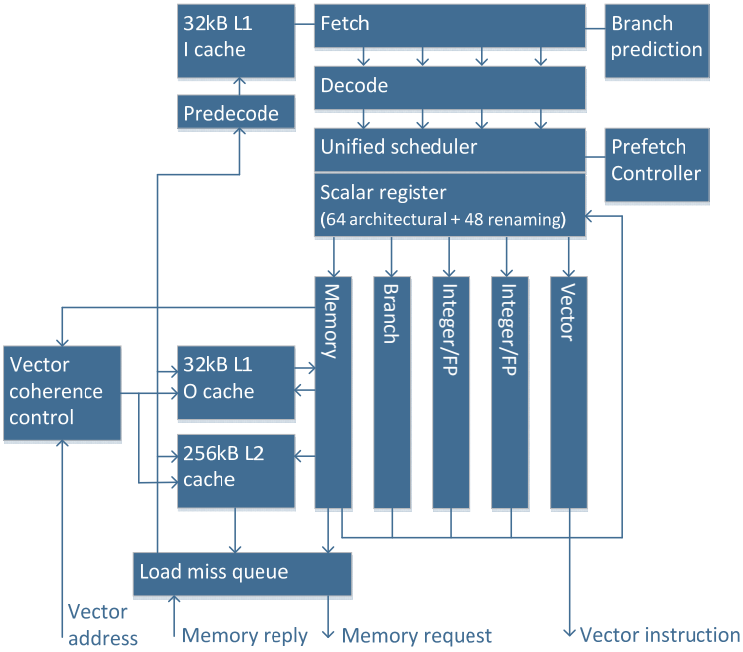


図 5. SPU ブロック図

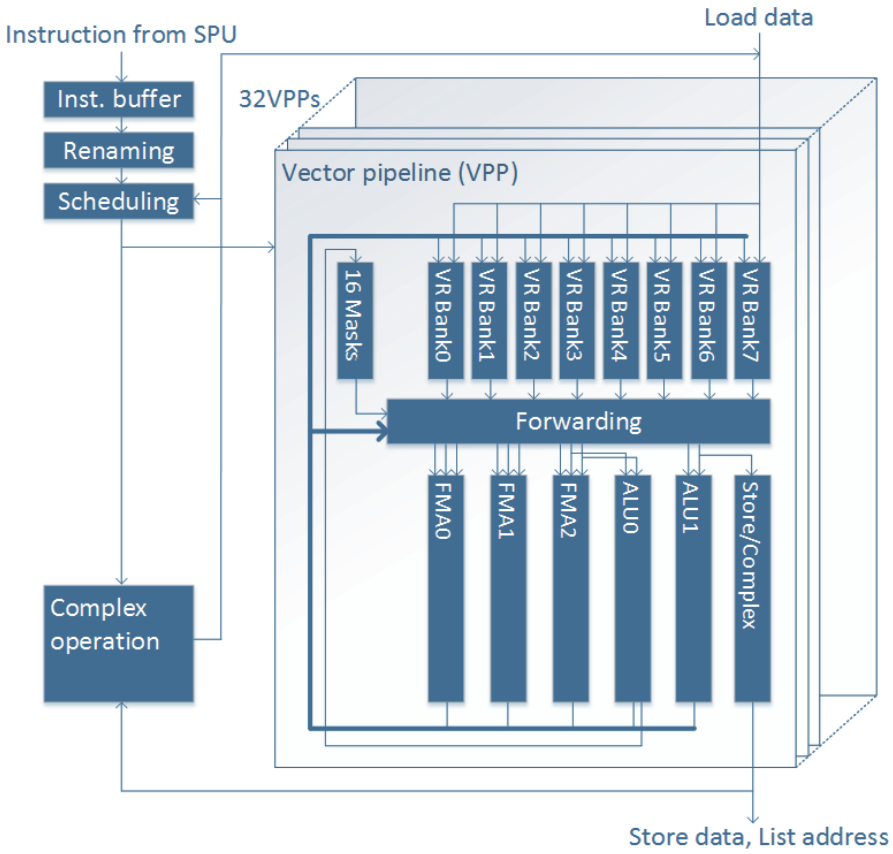


図 6. VPU ブロック図

CPU コアは、スカラ処理部(Scalar Processing Unit: SPU)、ベクトル処理部(Vector Processing Unit: VPU)の他、アドレス生成部、メモリネットワークによって構成されます。SPU は、すべての命令発行制御を行い、ベクトル命令やノード間通信命令を VPU や DMA エンジンへと発行します。1 サイクルに 4 命令の fetch/decode が可能で、VPU への専用パスを用いて 1 つのベクトル命令を発行可能です。32KB の L1 命令キャッシュ、32KB の L1 オペランドキャッシュ、256KB の L2 キャッシュを備えています。VPU は、32 本のベクトル・パイプラインから構成され、各パイプラインは最大で 3 つの 64bitFMA(浮動小数点積和演算)命令を同時に実行することが可能です。よって理論ベクトル演算性能は 32 パイプライン×3 セット×2(乗算＋加算)×1.6(GHz)=307.2GFlops となります。また、FMA 演算器は 32bit×2 セットの単精度 Packed 演算器としても動作することが可能であり、単精度浮動小数点演算性能は最大 714.4GFlops となります。各パイプラインは 1 サイクルに 64bit データのロードストアが可能であるため、コア辺りのメモリバンド幅は 8 バイト(64 ビット)×32 パイプライン×1.6GHz = 409.6GB/s(x2=ロード+ストア)となります。

コアあたりの論理ベクトルレジスタ数は 64 で、各ベクトルレジスタは 64bit の浮動小数点データを 256 要素格納できます。物理的には 256 本のベクトルレジスタを用意し、ベクトルレジスタのリネーミング機構を備えています。また、256 ビット長のマスクレジスタを 16 本備えています。

LLC は各コアで共有されるソフト制御可能なキャッシュメモリです。容量は 16MB で、ライトバック方式で制御されますが、SX-ACE の ADB(Assignable Data Buffer)に類似した機能として、ベクトル命令中のフラグによってデータ保持の優先度を指示することができます。

DMA エンジンは CPU コアと非同期に動作可能で、Vector Engine の主記憶の他、Vector Engine 内の各種レジスタ、Vector Host 内の主記憶にアクセス可能で、CPU コアだけでなく Vector Host の CPU からも利用可能です。

4.2 SX-Aurora TSUBASA B401-8

AOBA-A の構成要素(ノード)である SX-Aurora TSUBASA B401-8 の外観とブロック図をそれぞれ図 7 と図 8 に示します。B401-8 は Vector Host と 8 枚の Vector Engine で構成されます。Vector Host のプロセッサは AMD EPYC 7402P で、Vector Host あたり 1socket、24 コアを備えています。プロセッサの基本動作クロックは 2.8GHz で、シングルコアであれば最大で 3.35GHz までのブースト動作が可能です。プロセッサは 4 つの PCI Express switch と 2 つの InfiniBand HDR カードと直接接続されています。4 つの PCI Express switch はそれぞれ 2 つの Vector Engine カードと接続されています。プロセッサと InfiniBand HDR カードとの接続は PCI Express Gen4 x16 で、他の接続は PCI Express Gen3 x16 です。SX-Aurora TSUBASA B401-8 は Vector Host あたり 256GB の主記憶を備え、図 8 には図示されていませんが 960GB の SSD を備えています。

図 7. SX-Aurora TSUBASA B401-8 外観

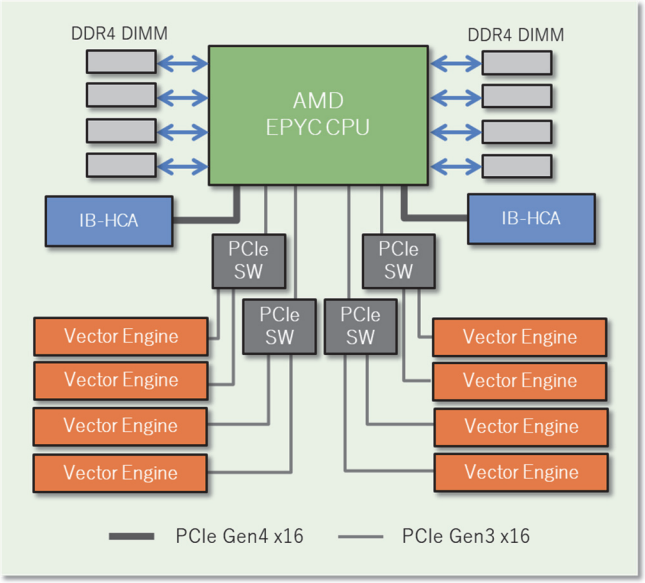


図 8. SX-Aurora TSUBASA B401-8 のブロック図

4.3 ノード間結合網

AOBA-A の 72 台の Vector Host は InfiniBand HDR による 2 段 Fat-Tree ネットワークによって接続されます。図 9 に InfiniBand ネットワーク構成図を示します。InfiniBand HDR スイッチとしては NVIDIA Mellanox QM8700 シリーズを利用します。Vector Host あたり 2 枚の HDR カードを介してエッジスイッチに接続されます。ノード間接続ネットワークは、AOBA-A だけでなく、AOBA-B、フロントエンドサーバなどの各種サーバやストレージを含めたシステム全体を、フルバイセクションバンド幅、ノンブロッキング構成により接続し、高速なデータ通信を可能とします。

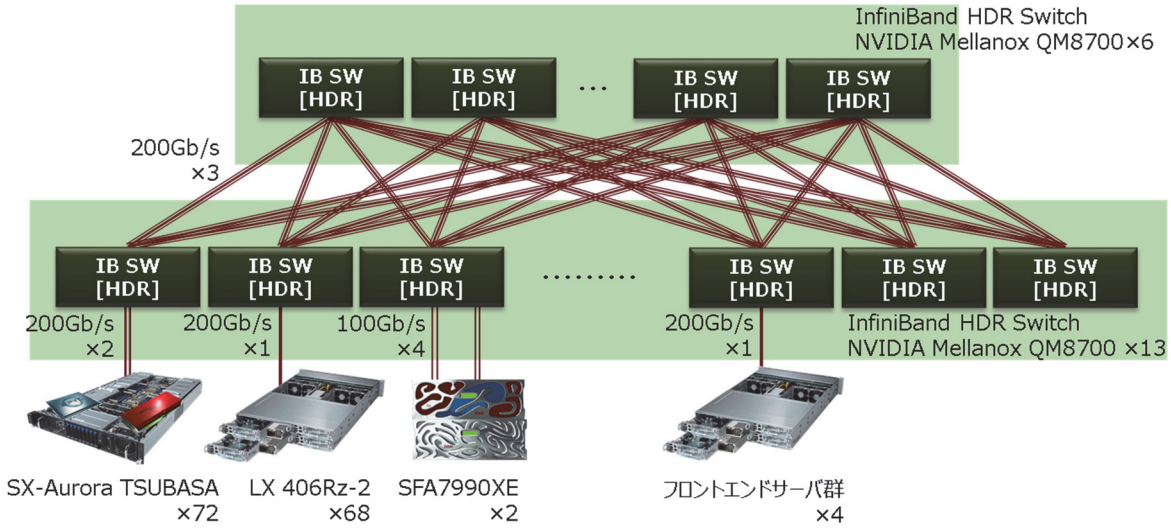


図 9. InfiniBand ネットワーク構成図

5. SX-Aurora TSUBASA のテクノロジ

5.1 LSI 技術

SX シリーズでは、従来から CMOS テクノロジによる高集積化を進め、SX-ACE では初めて 4 プロセッサを 1LSI に集積するなど、低消費電力と高性能の両立を目指した開発を行ってきました。AOBA-A の SX-Aurora TSUBASA システムに搭載している Vector Engine Type 20B では、最新の HBM2E メモリを利用することで LSI あたりのメモリバンド幅を 1.53TB/s に向上させています。LSI の外観を図 10 に、仕様を表 3 にまとめます。

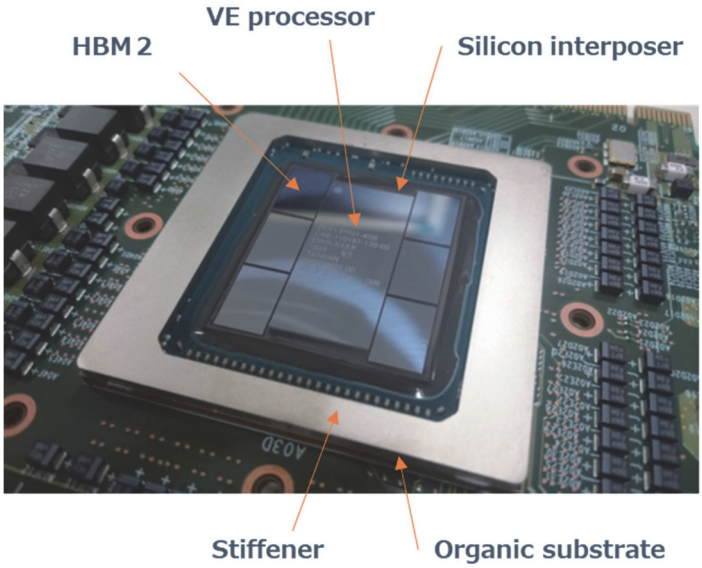


図 10. Vector Engine プロセッサ LSI 外観

表 3. CPU チップ諸元

テクノロジノード	16nm FinFET プロセス
Total gate count	274MGates
Total SRAM	283Mbits
コア電源電圧	0.89V
信号ピン数	10,676 IO total
ASIC(VE processor)サイズ	15mm×33mm
インタポーザサイズ	32.5mm×38mm
パッケージサイズ	60mm×60mm

5.2 実装/冷却技術

SX-Aurora TSUBASA B401-8 は、2U のユニットに 8 枚の Vector Engine カードを搭載しています。SX-Aurora TSUBASA B401-8 の水冷機構は in/out それぞれ 4 本の水冷ホースを備え、8 枚の Vector Engine カードと AMD EPYC 7402P 1 ソケットの計 9 つのプロセッサ LSI を冷却します。図 11 に B401-8 の内部構造図を、図 12 に B401-8 のラック搭載イメージを示します。

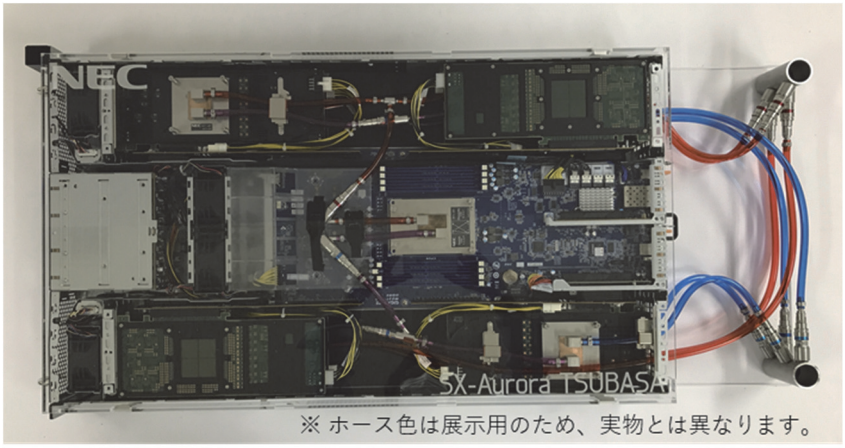


図 11. SX-Aurora TSUBASA B401-8 内部構造図

図 12. SX-Aurora TSUBASA B401-8 のラック搭載イメージ図

6. おわりに

このたび、AOBA-A に採用された SX-Aurora TSUBASA システムは、ラックあたりの演算性能を従来の SX-ACE システムと比較して 20 倍以上に高めるとともに、新しい Aurora アーキテクチャによって Linux OS 環境からベクトル型スーパーコンピュータをシームレスに利用できるようにしたことで、高性能と使いやすさの両立を実現することができました。この新しい SX-Aurora TSUBASA システム、そしてスーパーコンピュータ AOBA が、多くの方々の研究を前進させる一助になることと確信しています。NEC は今後も、様々な研究分野の発展を支えるため、高性能で使いやすいスーパーコンピュータの開発を継続していきます。

［大規模科学計算システム］

LX 406Rz-2 のハードウェア

横山 淳　　乾 重人　　水子 陽一
日本電気株式会社 AI プラットフォーム事業部

1. はじめに

東北大学サイバーサイエンスセンターで本年 10 月から運用を開始するスーパーコンピュータ AOBA は、576VE(72VH)のベクトルスーパーコンピュータ SX-Aurora TSUBASA(AOBA-A)と、68 ノードのスカラ処理向け大規模並列コンピュータ LX 406Rz-2(AOBA-B)が、InfiniBand HDR による高速インターコネクトで接続され、1.76PFlops の総合演算性能を有する最先端の大規模 HPC クラスタシステムです。

このうち、サブシステム AOBA-B で採用されている LX 406Rz-2 は、最新の AMD EPYC™ プロセッサを搭載し、4.096Tflops の高いノード性能を実現するとともに、InfiniBand HDR(200Gbps)をサポートした高速な通信性能を併せ持つ高密度実装サーバです。本稿では、LX 406Rz-2 のハードウェア、AMD EPYC™ 7002 シリーズプロセッサならびにインターコネクト技術について紹介します。

2. ハードウェア構成

2.1 LX 406Rz-2 の特長

LX 406Rz-2 は、米 AMD 社の最新プロセッサ EPYC™ 7002 シリーズを搭載し、2U ラックマウント型の筐体に 4 ノードを収容した、高性能、高効率、高密度な HPC クラスタ型計算サーバです。LX 406Rz-2 は、80PLUS® Titanium Level に対応した最大 2200W の大容量電源ユニットを筐体あたり 2 基搭載し、効率的で信頼性の高いシステムを実現しています。AOBA-B では、コア数が最も多く、電力と性能のバランスがとれた EPYC™ 7702 を採用しました。表 1 に LX 406Rz-2 の主要諸元を示します。本サーバの主な特長は以下のとおりです。

・高さ 2U の筐体に 4 ノード搭載し、ノード単位でホットプラグが可能
・ノードあたり AMD EPYC™ 7002 シリーズプロセッサを 2 基搭載
・DDR4-3200 対応の DIMM を 1 ノードあたり 256GB 搭載
・PCI Express Gen4(x16)に対応した拡張 IO スロットを 1 ノードあたり 2 スロット搭載
・InfiniBand HDR (200Gbps)をサポート
・IPMI2.0(Intelligent Platform Management interface 2.0)に対応し、遠隔地からシステムの環境監視、障害監視、障害ログの収集等が可能
・80PLUS Titanium に対応した 2200W の高効率電源を筐体あたり 2 基搭載

写真 1　LX 406Rz-2 正面

写真 2　LX 406Rz-2 背面

表 1　AOBA-B で採用されている LX 406Rz-2 主要諸元

項目		諸元
CPU (1 ノードあたり)	搭載 CPU	AMD EPYC™ 7702
	搭載数	2
	演算性能	4.096TFlops（倍精度浮動小数点演算）
メモリ (1 ノードあたり)	搭載メモリ	DDR4-3200、ECC 対応 RDIMM
	容量	256GB
	メモリバンド幅	409.60GB/s
内蔵ディスク(1 ノードあたり)		480GB SATA SSD
標準インターフェース(1 ノードあたり)		1x IPMI2.0
		2x USB3.0、1x VGA
拡張 IO スロット(1 ノードあたり)		2x PCI Express 4.0 (x16) 1x SIOM card
ノード間接続		InfiniBand HDR 200Gbps x1
筐体	外形寸法(HxWxD)	88mm(2U) x 447mm x 730mm
	質量	38.6kg
	電源	2x 2200W 80PLUS Titanium Level Power Supply

2.2 サーバの内部構成

図 1 に LX 406Rz-2 筐体のハードウェア構成を示します。LX 406Rz-2 は 2U シャーシの中に独立したノードを 4 台搭載しています。各ノードは電源ユニットとファンを、HDD バックプレーンを介して共有しています。各ノードと内蔵ディスク、電源ユニットはホットスワップが可能です。

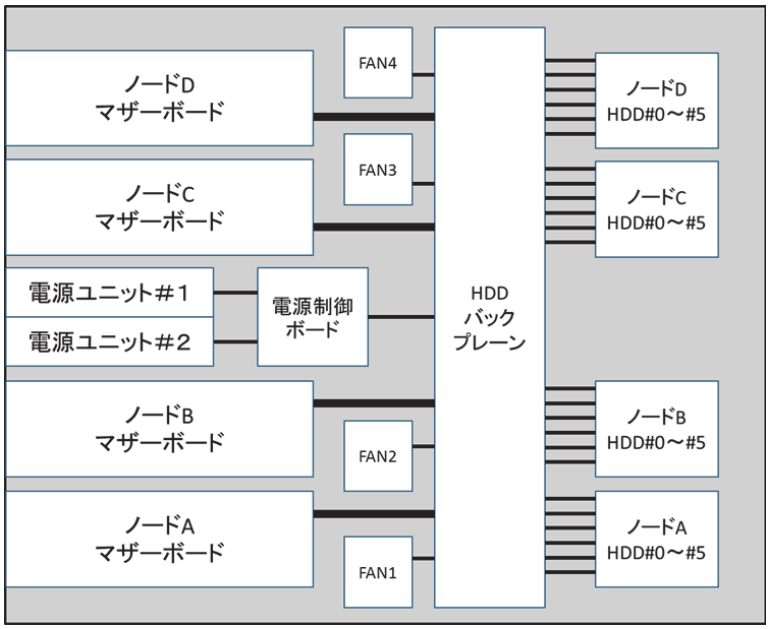


図 1　LX 406Rz-2 ハードウェア構成

図 2 にノードのブロック図を示します。

LX 406Rz-2 は、AMD EPYC™ 7002 シリーズプロセッサを 2 基搭載します。プロセッサ間は、4 チャネルの xGMI インターフェースで接続されています。また、プロセッサあたり 8 チャネルの DDR4-3200 対応メモリーインターフェスと、NEC のサーバとして初めて PCIe Gen4 に対応した拡張 PCI スロットを装備しています。初めて PCIe Gen4 に対応したことで、最速のインターコネクトである InfiniBand HDR(200Gbps)に対応することが可能になりました。LX 406Rz-2 には、サーバ管理機能として IPMI2.0 に対応したベースボード管理コントローラ(BMC)を搭載しています。これにより、温度、電力、ファンの動作状態、CPU やメモリのエラー監視、リモートからの電源のオン/オフなどを行うことが可能です。

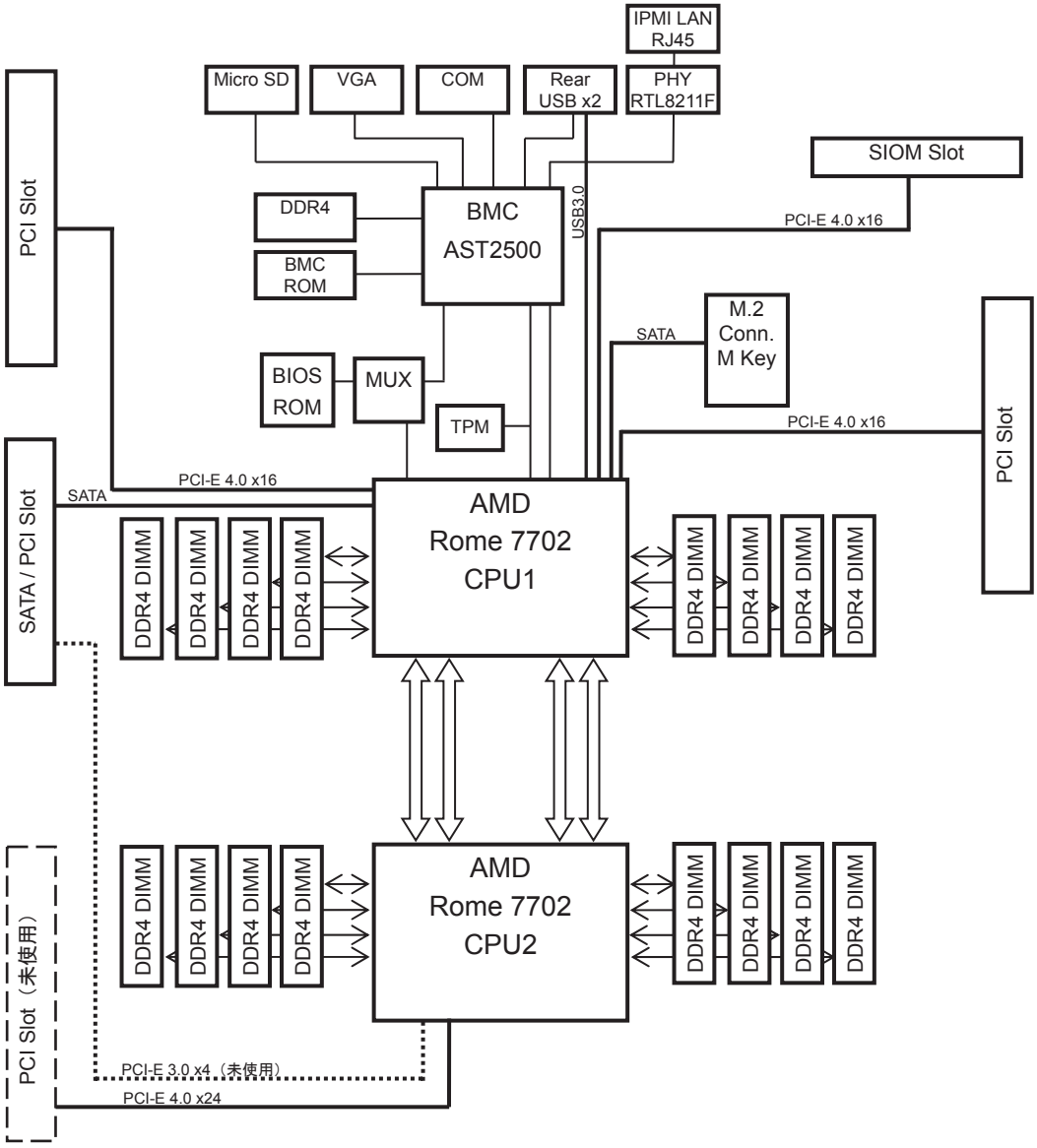


図 2　LX 406Rz-2 ノードブロック図

3. AMD EPYC™ 7002 シリーズ

3.1 AMD EPYC™ 7002 シリーズの特長

AMD EPYC™ 7002 シリーズ(コードネーム:Rome)は、2017 年に発売された第一世代である EPYC™ 7001 シリーズ(コードネーム:Naples)プロセッサの後継としてリリースされた製品です。第二世代である EPYC™7002 シリーズは、7nm プロセスで製造される CPU ダイ 8 個と 14nm プロセスで製造される I/O ダイ 1 つがパッケージ上に実装され、それらが高速インターコネクトで接続されています。CPU ダイには、最大 8 個の Zen2 コアとキャッシュメモリが搭載されています。I/O ダイには、8 チャネルのメモリコントローラと 128 レーンの PCI Express Gen4、SATA 等を制御する I/O コントローラが搭載されています。これにより、1 ソケットあたり最大 64 コアの CPU、8 チャネルの DDR4-3200 メモリ、最大 128 レーンの PCI Express Gen4 を備えています。

EPYC™ 7002 シリーズの Zen2 コアは、浮動小数点演算ユニットの強化、L3 キャッシュの容量拡大、CPU 間接続速度の向上などにより、第一世代の Zen コアと比較し IPC(Instruction Per Clock cycle)がおよそ 15%向上しています。

AMD EPYC™ 7002 シリーズプロセッサは、サーバ向けのプロセッサとして初めて PCI Express Gen4 をサポートしています。これにより、第一世代の EPYC™ 7001 シリーズと比較して、2 倍の I/O 性能を実現し、HPC クラスタシステムを接続するインターコネクトの帯域幅を 2 倍に向上しています。
EPYC™ 7002 シリーズの主な特長は以下の通りです。

・7nm プロセスの「Zen2」アーキテクチャを採用
・ソケットあたり最大 64 コア 128 スレッド
・DDR4-3200 メモリに対応し、ソケットあたり最大 4TB のメモリを搭載可能
・サーバプロセッサとして初めて PCI Express Gen4 をサポート
・第一世代と比較しソケット当たりの理論ピーク性能(FLOPS)をおよそ 4 倍に向上

3.2 AMD EPYC™ 7702 諸元

AOBA-B の LX 406Rz-2 では、64 個のコアをもち、性能と電力のバランスが取れた EPYC™ 7702 を採用しました。表 2 に AMD EPYC™ 7702 の諸元を示します。

表 2 AMD EPYC™ 7702 諸元

項目	諸元
コア数/スレッド数	64/128
周波数	2.0GHz
消費電力(TDP)	200W
製造プロセスルール	7nm
メモリーインターフェス	DDR4-3200
メモリチャネル数	8
最大メモリ容量	4TB
最大メモリ帯域幅	204.8GB/s
PCI Express	Gen4 128 レーン

4. インターコネクト

4.1 インターコネクトの特長

スーパーコンピュータ AOBA のインターコネクトは、InfiniBand HDR による 2 段 Fat-Tree ネットワークを採用し、フルバイセクションバンド幅、ノンブロッキング構成によりすべての演算サーバ、ストレージ、フロントエンドサーバで高速なデータ通信を可能にしています。

スイッチは NVIDIA Mellanox QM8700 シリーズを採用し、6 台のコアスイッチと 13 台のエッジスイッチでノード間接続ネットワークを構成しています。これらのスイッチにはマネージドスイッチを採用しており、うち 2 台のスイッチはサブネットマネージャとして動作し、ファブリック内のホストや通信経路の検索、監視、ならびに障害検出・回復などを行います。

2 台のサブネットマネージャは冗長化されており、1 台がプライマリサブネットマネージャ、もう 1 台がセカンダリサブネットマネージャとして動作します。プライマリサブネットマネージャで障害が発生すると、経路情報などをセカンダリサブネットマネージャが引き継ぎ動作を継続します。このように、スーパーコンピュータ AOBA のインターコネクトは、信頼性の高い高速ネットワークを実現しています。

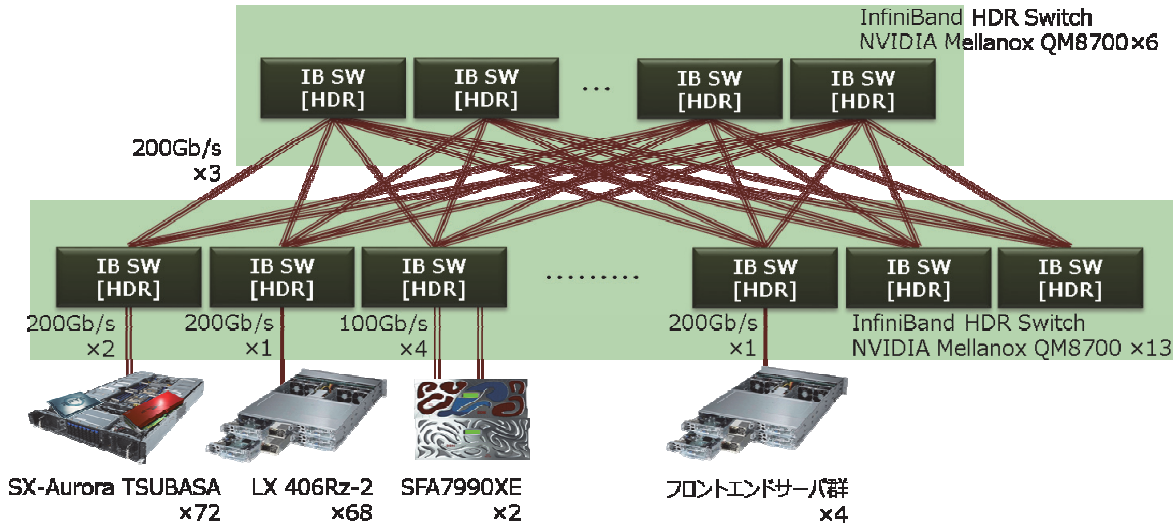


図 3　InfiniBand ネットワーク図

4.2 InfiniBand HDR スイッチ（QM8700 シリーズ）

QM8700 シリーズは、1 筐体に 40 ポート、16Tbps のノンブロッキング帯域幅と、90ns 以下のポート間レイテンシを実現します。LX 406Rz-2 に搭載する InfiniBand HDR(200Gbps) HCA カードと組み合わせることで最高のファブリック性能を実現します。表 3 に QM8700 シリーズの主要諸元を示します。

表 3　QM8700 シリーズ 主要諸元

項目		諸元
筐体		19 インチラック対応 1U 筐体
ポート数		HDR (200Gbps)　40 ポート HDR100 (100Gbps) 80 ポート (スプリッタケーブル使用時)
ファブリック管理機能		On-board Subnet Manager (SM) Unified Fabric Manager (UFM) agent
運用管理	管理ポート	100/1000 RJ45 Ethernet ポート RS232 コンソールポート USB ポート
	管理インターフェース	CLI、WebUI、SNMP v1/v2/v3
電源		2x ホットスワップ可能 冗長電源モジュール
FAN		ホットスワップ可能ファンモジュール

5. むすび

本稿では、AOBA-B で採用されている HPC クラスタ型計算サーバ LX 406Rz-2 の主要な機能、特長についてご紹介しました。

科学技術計算など高い性能を追求する HPC 領域では、設置スペースや消費電力等の様々な問題に向き合いながら、高い処理能力が求められます。NEC の HPC クラスタソリューションは、さまざまなワークロードに適用した高い電力性能比を有するCPUと高密度実装サーバにより、お客様の課題に応えて参ります。

［大規模科学計算システム］

利用申請と利用負担金について

情報部情報基盤課 共同利用支援係

1　はじめに

サイバーサイエンスセンター (以下「本センター」という。) は，研究，教育等に係る情報化 を推進するための実践的調査研究，基盤となる設備等の整備および提供その他必要な専門的業務を行う 全国共同利用の施設です。本センターは大規模科学計算システムを設置し，本センターの前身の一つである旧大型計算機センターの機能と知識の蓄積を継承して，最先端の大規模科学技術計算環境および高度利用環境の提供，並びに利用者への技術的支援を行っています。

2　利用形態

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には以下の利用形態があります。利用形態についてはそれぞれの項目を参照してください。

- 大学・学術利用 → 項目 4.2
- 民間企業利用 → 項目 4.3
- センターとの共同研究 → 項目 5.1
- 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） → 項目 5.2
- 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） → 項目 5.3

3　利用負担金

3.1　利用負担金について

利用負担金には演算負担経費，ファイル負担経費および出力負担経費の 3 つがあります（表 1，2，3)。スーパーコンピュータと並列コンピュータを利用すると，演算負担経費が発生します。

共有利用（従量）は利用する VE 数（スーパーコンピュータの場合）もしくはノード数（並列コンピュータの場合）と経過時間によって負担額が決定します。

共有利用（定額）は負担額を先払いし，課金対象時間の合計に相当する時間までそれぞれのコンピュータが利用が出来ます。負担額を追加することも可能です。年間の利用時間が課金対象時間の合計に満たない場合，負担額の返金はありません。

占有利用は VE またはノードを占有して確保しますので，他の利用者のジョブが終了するのを待つ必要がありません。

利用者が負担経費を直接支払うのは，大学・学術利用（項目 4.2)，民間企業（項目 4.3）およびセンターとの共同研究（項目 5.1）です。センターとの共同研究では，演算負担金の割引制度が適用されます。

請求書は半期（6 ヶ月）ごとに，利用者を取りまとめている支払責任者に発行します。利用期間と利用負担金の請求次期については項目 3.2 をご参照ください。

■大学・学術利用　表 1 が適用されます。利用者が学術利用に該当するかは共同利用支援係にお問合せください。

■民間企業利用　成果公開型の利用については表 2 が，成果非公開型の利用については表 3 が適用されます。

表 1　基本利用負担金【大学・学術利用】

区分	項目	利用形態	負担額及び課金対象時間
演算負担経費	スーパーコンピュータ	共有（無料）	利用 VE 数 1（実行数，実行時間の制限有）　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 1 時間につき　125 円
		共有（定額）	負担額 10 万円につき課金対象時間 800 時間分使用可能
		占有	利用 VE 数 8　利用期間 3 ヶ月につき　270,000 円
	並列コンピュータ	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 1 時間につき　22 円
		共有（定額）	負担額 10 万円につき課金対象時間 4,600 時間分使用可能
		占有	利用ノード数 1　利用期間 3 ヶ月につき　47,000 円
ファイル負担経費		共有	5TB まで無料，追加容量 1TB につき年額　3,000 円
		占有	10TB まで無料，追加容量 1TB につき年額　3,000 円
出力負担経費	大判プリンタによるカラープリント		フォト光沢用紙 1 枚につき　600 円 クロス紙 1 枚につき　1,200 円

備考

1. 負担額が無料となるのは専用のキューで実行されたものとし，制限時間を超えた場合は強制終了する。
2. 演算負担経費の課金対象時間については半期毎 (4 月から 9 月及び 10 月から 3 月) に合計し，1 時間未満を切上げて負担金を請求する。
3. 演算負担経費について定額制を選択した場合はスーパーコンピュータ及び並列コンピュータを課金対象時間の範囲内で共用できる。
4. 占有利用期間は年度を超えないものとし，期間中に障害，メンテナンス作業が発生した場合においても，原則利用期間の延長はしない。
5. ファイル負担経費については申請日から当該年度末までの料金とする。運用期間が 1 年に満たない場合は，月割りをもって計算した額とする。

表 2　基本利用負担金【民間企業利用（成果公開型)】

区分	項目	利用形態	負担額及び課金対象時間
演算 負担経費	スーパー コンピュータ	共有 （無料）	利用 VE 数 1（実行数，実行時間の制限有） 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 1 時間につき　250 円
		共有 （定額）	負担額 20 万円につき課金対象時間 800 時間分使用可能
		占有	利用 VE 数 8　利用期間 3 ヶ月につき　540,000 円
	並列 コンピュータ	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 1 時間につき　44 円
		共有 （定額）	負担額 20 万円につき課金対象時間 4,600 時間分使用可能
		占有	利用ノード数 1　利用期間 3 ヶ月につき　94,000 円
ファイル 負担経費		共有	5TB まで無料，追加容量 1TB につき年額　6,000 円
		占有	10TB まで無料，追加容量 1TB につき年額　6,000 円
出力 負担経費	大判プリンタによる カラープリント		フォト光沢用紙 1 枚につき　1,200 円 クロス紙 1 枚につき　2,400 円

備考

1. 負担額が無料となるのは専用のキューで実行されたものとし，制限時間を超えた場合は強制終了する。
2. 演算負担経費の課金対象時間については半期毎 (4 月から 9 月及び 10 月から 3 月) に合計し，1 時間未満を切上げて負担金を請求する。
3. 演算負担経費について定額制を選択した場合はスーパーコンピュータ及び並列コンピュータを課金対象時間の範囲内で共用できる。
4. 占有利用期間は年度を超えないものとし，期間中に障害，メンテナンス作業が発生した場合においても，原則利用期間の延長はしない。
5. ファイル負担経費については申請日から当該年度末までの料金とする。運用期間が 1 年に満たない場合は，月割りをもって計算した額とする。

表 3　基本利用負担金【民間企業利用（成果非公開型)】

区分	項目	利用形態	負担額及び課金対象時間
演算 負担経費	スーパー コンピュータ	共有 （無料）	利用 VE 数 1（実行数，実行時間の制限有） 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 1 時間につき　500 円
		共有 （定額）	負担額 40 万円につき課金対象時間 800 時間分使用可能
		占有	利用 VE 数 8　利用期間 3 ヶ月につき　1,080,000 円
	並列 コンピュータ	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 1 時間につき　88 円
		共有 （定額）	負担額 40 万円につき課金対象時間 4,600 時間分使用可能
		占有	利用ノード数 1　利用期間 3 ヶ月につき　188,000 円
ファイル 負担経費		共有	5TB まで無料，追加容量 1TB につき年額　12,000 円
		占有	10TB まで無料，追加容量 1TB につき年額　12,000 円
出力 負担経費	大判プリンタによる カラープリント		フォト光沢用紙 1 枚につき　2,400 円 クロス紙 1 枚につき　4,800 円

備考

1. 負担額が無料となるのは専用のキューで実行されたものとし，制限時間を超えた場合は強制終了する。
2. 演算負担経費の課金対象時間については半期毎 (4 月から 9 月及び 10 月から 3 月) に合計し，1 時間未満を切上げて負担金を請求する。
3. 演算負担経費について定額制を選択した場合はスーパーコンピュータ及び並列コンピュータを課金対象時間の範囲内で共用できる。
4. 占有利用期間は年度を超えないものとし，期間中に障害，メンテナンス作業が発生した場合においても，原則利用期間の延長はしない。
5. ファイル負担経費については申請日から当該年度末までの料金とする。運用期間が 1 年に満たない場合は，月割りをもって計算した額とする。

3.2　利用負担金の請求

負担金の請求は半年ごとに行います（表 4)。前期の 5,000 円未満の請求は後期に繰り越します。後期の 1,000 円未満の請求は行いません。

学外の方については，特に支払費目名の入った利用負担金請求書を希望する場合や，請求書の適要欄等について不明な点がある場合は，請求書発行前（9，3 月末）までに 会計係（022-795-3405）へご連絡下さい。また，通常の請求時期以外に請求書の発行が必要な場合は，共同利用支援係（022-795-6251）へご連絡下さい。

表 4　利用期間と請求時期

	利用期間	請求書発行
前期	4 月〜9 月	10 月上旬
後期	10 月〜3 月	4 月上旬

4　利用申請

4.1　利用資格

利用できるのは，次のいずれかに該当する方です。大規模科学計算システムの利用は，学術研究及び教育等を目的とするものに限られます。（社会貢献の一環としての民間等外部機関の利用を含みます。）

- 大学，短期大学，高等専門学校及び大学共同利用機関の教員及び学生
- 文部科学省所轄機関及び文部科学大臣が主務大臣である独立行政法人の研究職員
- 学術研究を目的とする研究機関で，東北大学サイバーサイエンスセンター長 (以下「センター長」という。) が認めた機関に所属し，専ら研究に従事する者
- 文部科学省及び独立行政法人日本学術振興会所管の科学研究費補助金で研究を行う者
- 国及び地方公共団体より委託 (受託) を受けた研究を行う者
- 前に掲げる者のほか，特にセンター長が認めた者

■内規

- 東北大学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利用に関する内規
  （https://www.ss.cc.tohoku.ac.jp/pdf/システムの利用に関する内規.pdf）
- 東北大学サイバーサイエンスセンター大規模科学計算システム利用負担金内規
  （https://www.ss.cc.tohoku.ac.jp/pdf/利用負担金内規.pdf）
- 東北大学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民間機関等利用内規
  （https://www.ss.cc.tohoku.ac.jp/pdf/民間機関等利用内規.pdf）

4.2　大学・学術利用

4.2.1　利用の手続

大規模科学計算システムを利用するためには利用者登録が必要です。

■利用申請　利用申請書（https://www.ss.cc.tohoku.ac.jp/word/利用申請書.doc）に必要事項を記入のうえ，共同利用支援係に提出してください。後日，利用者番号（ログイン ID）と初期パスワードを記載した「システム利用承認書」を送付します。

■変更　申請内容に変更が生じた場合は，利用変更届（https://www.ss.cc.tohoku.ac.jp/word/利用変更届.doc）に必要事項を記入のうえ，共同利用支援係に提出してください。

■取消　次年度以降も自動継続され利用できます。利用を取り止める場合は，利用取消届（https://www.ss.cc.tohoku.ac.jp/word/利用取り消し届.doc）に必要事項を記入のうえ，共同利用支援係に提出してください。

利用変更届，利用取り消し届に関しては申請用紙を使わず、内容をメール本文に記載していただいても構いません。

4.2.2　外国人利用者についてのお願い（指導教員，受入教員向け）

外国人がスーパーコンピューターシステムの利用申請を行う場合にあっては，下記のリンク先の説明を踏まえ申込を行うようにして下さい。

- 「東北大学における安全保障輸出管理（スーパーコンピュータ利用）」
  （http://www.bureau.tohoku.ac.jp/export/supakonriyou.html）

また利用申請には以下の書類の提出が必用です。ダウンロードしてご利用ください。

- 居住性チェックリスト（日本語版）
  （https://www.ss.cc.tohoku.ac.jp/word/check_list.docx）
- 居住性チェックリスト（英語版）
  （https://www.ss.cc.tohoku.ac.jp/word/check_list_eng.docx）

4.3　民間企業利用

本センターでは社会貢献の一環として，大学で開発された応用ソフトウェアとスーパーコンピュータ利用の民間企業への提供を実施いたします。民間企業利用サービスについての詳細は，センターウェブサイト「民間企業利用」（https://www.ss.cc.tohoku.ac.jp/business/）および「東北大学サイバーサイエンスセンター大規模計算システム民間企業利用サービス利用課題募集要項」（https://www.ss.cc.tohoku.ac.jp/pdf/民間企業利用サービス募集要項.pdf）をご覧ください。

4.3.1　利用申請

応募，利用，利用終了時には以下の書類の提出が必要となります。

- 民間企業利用サービス課題申込書
  （https://www.ss.cc.tohoku.ac.jp/word/民間企業利用サービス課題申込書.doc）
- 民間企業利用誓約書
  （https://www.ss.cc.tohoku.ac.jp/word/民間企業利用サービス誓約書.doc）
- 民間企業利用サービス報告書
  （https://www.ss.cc.tohoku.ac.jp/word/民間企業利用サービス報告書.doc）

4.3.2　利用課題区分

募集を行う利用課題区分は以下の 2 つです。

■大規模計算利用　ライフサイエンス，もの作り技術，情報通信，環境エネルギー，社会基盤の課題分野をはじめとして，様々な分野における特に優れた課題を募集します。大学で開発された先端的シミュレーションソフトウェアとスーパーコンピュータの利用課題を民間企業から募り，企業によるイノベーション創出を支援します。利用成果公開型と非公開型の利用があり、目標設定を行った上で、1 年間を一区切りとして利用期間を決定いたします。課題終了時に継続利用の申請を行うことが可能です。有償での利用となり民間企業用の負担金制度を適用します。

■トライアルユース　新しいニーズを掘り起こしイノベーション創出につながる利用課題を民間企業から募ります。 分野は特定しません。産業界の潜在的高性能計算需要を掘り起こし，大学で開発された先端的シミュレーションソフトウェアとスーパーコンピュータを有償で利用して頂く下地を作ることが目的となります。先端研究施設共用促進事業での利用など，これまでに本センターを利用したことがない民間企業を募ります。利用成果は公開を原則とし，最大 1 カ月間を無料で利用することが可能です。

4.3.3　成果公開型／成果非公開型

成果公開型は利用成果を含めた利用サービス報告書を利用終了から 30 日以内に提出することが必要です。また 1 年以内に SENAC に成果の記事を投稿することが必要です。成果公開型で利用している企業名と研究内容およびその成果について，センターが一般向けに公開することが出来ます。利用負担額は大学・学術利用の 2 倍です。

成果非公開型は利用成果や利用内容についての報告は必要ありませんが，利用サービス報告書の提出が必要です。成果非公開型で利用している企業名について，センターはセンター内部の会議に報告することが出来ます。利用負担額は大学・学術利用の 4 倍です。

4.3.4　応募資格

以下の全ての項目を満たすことが必要です。

- 日本国内で利用がなされること
- 東北大学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利用に関する内規に従うこと
- 採択課題の目的にのみ利用すること
- 平和利用のみに限ること
- 人権および利益保護への配慮を行うこと
- 文部科学省「生命倫理・安全に対する取組」に適合すること
- 経済産業省「安全保障貿易管理について」に適合すること
- 課題終了後，利用報告書を速やかに報告すること

4.3.5　利用可能なソフトウェア

- TAS-code (Tohoku University Aerodynamic Simulation Code)
- 数値タービンシミュレーション
- 超臨界流体シミュレータ
- Gaussian

5　センターとの共同研究・JHPCN・HPCI

5.1　センターとの共同研究

サイバーサイエンスセンターでは，研究者のより良いスーパーコンピューティング環境を構築するために，スーパーコンピューティングに関する共同研究の募集を行っています。

5.1.1　応募者の資格

本センター大規模科学計算システムの利用有資格者

5.1.2　応募期間，応募方法

別ニュース記事を参照

5.1.3　共同研究の要件

1. プログラムのベクトル化や並列化の研究を，本センターのスタッフと共同で行います。
2. 共同研究を行うためにプログラムコード等を本センターに提供していただきます。
3. 共同研究の研究成果，あるいは研究成果であるプログラムにより得られた結果を論文等で公表する際，共同研究をふまえて本センターの貢献を明記してください（例えば，共著者，謝辞等）。
4. 年度末に共同研究実績報告書（所定の様式）を提出していただきます。この報告書にも本センターの貢献を必ず明記してください。
5. 共同研究終了から 1 年以内に本センター大規模科学計算システム広報誌 SENAC に共同研究の成果を投稿していただきます。この記事には，プログラムのベクトル化や並列化などの具体的な内容と効果，及び，本センターとの共同した取組みを記述していただく必要があります。
6. 研究成果を学会等へ報告した場合は，その別刷等を本センターに提出してください。
7. 課題 [A] の採択課題の中から JHPCN 萌芽型共同研究課題が採択されます。JHPCN 萌芽型共同研究課題の採択者には，JHPCN 主催のシンポジウムでの発表等を依頼する場合があります。

5.1.4　助成内容等

1. 課題 [A] は，演算負担経費（全額）の 2/3 を本センターが負担します。
2. 課題 [B] は，20 万円を超えた分の演算負担経費の 2/3 を本センターが負担します。
3. 本センターの負担金額の上限は，課題 [A][B] ともに 500 万円とします。
4. 助成対象は共有利用による演算負担経費とし，占有利用は助成対象外とします。

助成内容についての詳細は、最新のニュース記事をご参照ください。

5.2　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）

サイバーサイエンスセンターは「学際大規模情報基盤共同利用・共同研究拠点」として文部科学大臣の認定を受け，活動を行っております。「学際大規模情報基盤共同利用・共同研究拠点」は，北海道大学，東北大学，東京大学，東京工業大学，名古屋大学，京都大学，大阪大学，九州大学にそれぞれ附置するスーパーコンピュータを持つ８つの共同利用の施設を構成拠点とし，各センターからなる「ネットワーク型」共同利用・共同研究拠点として，平成 22 年 4 月より本格的に活動を開始しました。

本ネットワーク型拠点の目的は，超大規模計算機と大容量のストレージおよびネットワークなどの情報基盤を用いて，地球環境，エネルギー，物質材料，ゲノム情報，Web データ，学術情報，センサーネッ

トワークからの時系列データ，映像データ，プログラム解析，その他情報処理一般の分野における，これまでに解決や解明が極めて困難とされてきた，いわゆるグランドチャレンジ的な問題について，学際的な共同利用・共同研究を実施することにより，我が国の学術・研究基盤の更なる高度化と恒常的な発展に資することにあります。本ネットワーク型拠点には上記の分野における多数の先導的研究者が在籍しており，これらの研究者との共同研究によって，研究テーマの一層の発展が期待できます。

申し込み方法等詳細につきましては JHPCN のウェブサイトをご覧ください。

https://jhpcn-kyoten.itc.u-tokyo.ac.jp/

5.2.1　サイバーサイエンスセンターの取組

東北大学サイバーサイエンスセンターでは，ベクトル型とスカラ型二つのスーパーコンピュータを運用することで，多様化の進むユーザ・アプリケーションの要求に柔軟に対応できる計算環境を提供しています。

また，教員と技術系職員が連携して，当センターを利用する共同研究を実施する体制を整備し，プログラムの高度化等で計算科学者と計算機科学者が密に連携し，計算機アーキテクチャ，高性能基盤ソフトウェア，高性能計算技術に関する研究を遂行するなど，高性能計算機を用いた科学の進展，イノベーションの創生に向け取り組んでいます。

5.3　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）

HPCI とは，「富岳」と全国の主要なスーパーコンピュータをネットワークでつなぎ，多様なニーズに応える革新的な計算機環境を実現するための基盤システムです。東北大学サイバーサイエンスセンターは HPCI システムの構成機関として参画しております。

5.3.1　HPCI の目的（HPCI ポータルサイトより）

- 個別の計算資源提供機関ごとに分断されがちな全国の幅広いハイパフォーマンスコンピューティング（HPC）ユーザー層が全国の HPC リソースを効率よく利用できる体制と仕組みを整備し提供すること。
- 全国規模でニーズとリソースのマッチングを可能とすることにより，萌芽的研究から大規模研究まで，また産業利用にわたる幅広い HPC 活用を加速するとともに，計算科学コミュニティを醸成・拡大し，成果の社会還元を図ること。

5.3.2　課題の申請方法

「富岳」および「富岳」以外の HPCI システム共用計算資源の一般利用は，原則として年 1 回公募されます。ただし，産業界からの利用を促進するため，産業利用のトライアル・ユースと成果を非公開とする有償利用（個別利用）は年間を通じて随時受け付けられます。 利用を希望される方は利用研究課題を申請し，審査を経て利用が可能となります。

申し込み方法等詳細につきましては HPCI のウェブサイトをご覧ください。

https://www.hpci-office.jp/

6　利用負担金の確認方法

6.1　プロジェクトコードの確認

利用負担金はプロジェクトコード毎に合算されます。プロジェクトコードの名称変更，追加などについては共同利用支援係までお問い合わせください。

6.2　課金状況の確認

フロントエンドサーバ上で，pkakin コマンド（プロジェクトコード毎の課金状況），ukakin コマンド（利用者番号毎の課金状況）が確認できます。

6.3　ジャーナル（利用明細）の確認

フロントエンドサーバ上で，plist コマンド（プロジェクトコード毎のジャーナル一覧），ulist コマンド（利用者番号毎のジャーナル一覧）が確認できます。

7　研究成果の提出

本センターでは，学術研究を支える世界最高水準の大規模科学計算システムの導入と利用環境の整備・拡充を行い，研究の発展に資することを心掛けております。今後もシステムの整備を進めていくには，大規模科学計算システムが多くの研究分野で必要不可欠であり，かつ研究成果が得られていることを広くアピールしていく必要があります。

このため利用者の皆様には，本センター大規模科学計算システムを利用して得られた研究成果の一覧をご提出くださいますようお願いいたします。

研究成果一欄　：著者名，論文名，掲載誌（巻号頁），発表年
提出方法　　：電子メールでお願いします。email: seika[at]cc.tohoku.ac.jp
締切り日　　：翌年度 4 月中旬
問合せ先　　：共同利用支援係 uketuke[at]cc.tohoku.ac.jp

また論文等を発表される際には，本センターを利用した旨を明記してくださるようお願いいたします。

■記入例　「本研究の実験結果の一部は，東北大学サイバーサイエンスセンター大規模科学計算システムを利用して得られた。」

■Example　Part of the experimental results in this research were obtained using supercomputing resources at Cyberscience Center, Tohoku University.

8　問い合わせ先

利用申請と利用負担金についてのお問い合わせは共同利用支援係までお願いいたします。

■共同利用支援係：　uketuke[at]cc.tohoku.ac.jp

ご不明な点，ご質問等ございましたら，お気軽にセンターまでお問い合わせください。問い合わせについては「利用相談」(https://www.ss.cc.tohoku.ac.jp/consultation/) をご参照下さい。

［大規模科学計算システム］

鍵ペアの作成とログイン方法

情報部情報基盤課 共同研究支援係 共同利用支援係

1. はじめに

本センターのシステムは、セキュリティ対策として、公開鍵認証方式による SSH 接続を採用しています。また、フロントエンドサーバは、ログインサーバを経由しなければログインできない構成としています。

本稿では、公開鍵認証方式で使用する鍵ペアの作成と各サーバのログイン方法についてご紹介します。解説では以下のターミナル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（4 章）

以前のシステム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。5 章からお読みください。

2. ログイン認証方式

表 1 に、各サーバのログイン認証方式を示します。

表 1　各サーバのログイン認証方式

サーバ名	用途	ログインホスト名	認証方式
ログインサーバ	フロントエンドサーバの入口（踏み台サーバ）	login.cc.tohoku.ac.jp	公開鍵
フロントエンドサーバ	計算機の利用（コンパイル、ジョブ実行、等）	(※1)	公開鍵またはパスワード
データ転送サーバ	ストレージシステムとの大容量のデータ転送	file.cc.tohoku.ac.jp	公開鍵
HPCI 用ログインノード	HPCI、HPCI-JHPCN ユーザ専用ログインノード	hpcif.cc.tohoku.ac.jp	公開鍵
-	センター内施設の利用(※2)	-	パスワード

（※1）フロントエンドサーバは、ログインサーバからしかログインできません。本稿では多段 SSH による接続方法を解説します。

（※2）本センター内の施設（大判カラープリンタ、利用者端末、講習会端末）はパスワード認証でご利用いただけます。利用にあたり、秘密鍵を持参する必要はありません。

3.　鍵ペアの作成からログインまでの流れ

図 1 に、鍵ペア作成からログインまでの流れを示します。①は 4 章、②③は 5 章で詳しく解説します。

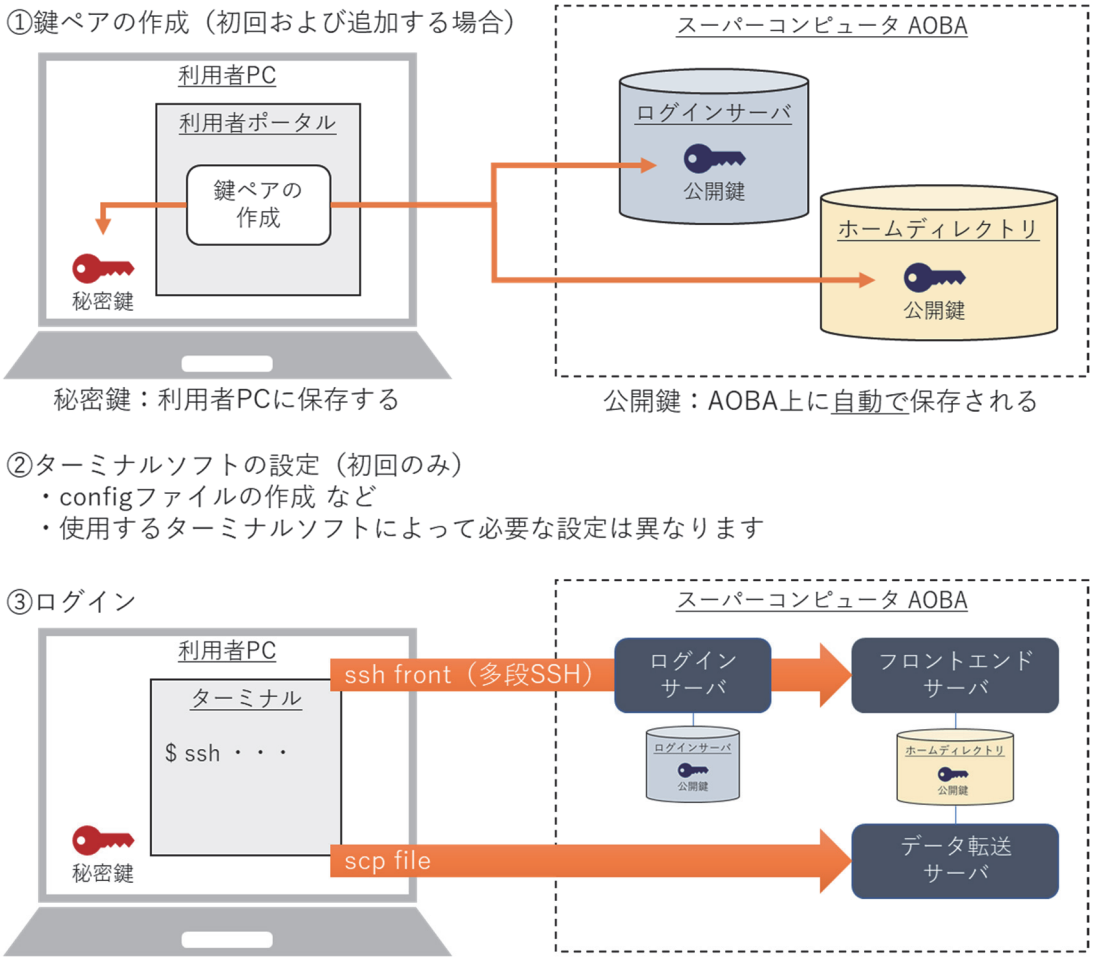


図 1　鍵ペア作成からログインまでの流れ

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。

② ターミナルソフトの設定（初回ログイン時）

各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は異なります。

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ログインします。フロントエンドサーバは、ログインサーバを経由して多段 SSH でログインします。

4. 公開鍵認証方式で使用する鍵ペアの作成

4.1. 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、等）のリスクが非常に高まり、大変危険です。ご注意願います。

- 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ノード上に保存しないこと

4.2. 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）

○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 先から利用者ポータルを開きます。
利用者ポータルには、利用者番号とパスワード（※）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¥.ssh
（macOS／Linux の場合）　~/.ssh
公開鍵は、ホームディレクトリ（~/.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パスワードの変更方法は、6 章を参照してください。

○別の PC からログインする場合（ログイン端末を追加する場合）

既存の秘密鍵を使いまわすのではなく、ログイン端末ごとに鍵ペアを作成してください。初回ログイン時と同じ手順で、新しい鍵ペアを追加します。

5. 各サーバのログイン方法

5.1. ターミナルソフトの設定（初回ログイン時）

利用者のローカル PC 上で、ターミナルソフトの設定を行います。

以降の解説は、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¥.ssh
（macOS／Linux の場合）　~/.ssh

各ログインホストのホスト名は、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（他の設定との重複は不可）。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。

（ログインサーバ）　login
（フロントエンドサーバ）　front
（データ転送サーバ）　file
（HPCI 用ログインノード）　hpcif

(1) macOS／Linux の場合は、秘密鍵のパーミッションの変更（600 に設定）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。

```
$ chmod 600 ~/.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。拡張子はつけません。
（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、ログインできません。ご注意ください）

(3) config ファイルに以下の設定を記述します。**太字下線**の部分は、ご自身の環境に合わせて読み替えてください。

○フロントエンドサーバを利用するための設定（※）

```
# ログインサーバの設定（ホスト名を”login”とする場合）
Host login                          # ホスト名を指定
HostName login.cc.tohoku.ac.jp      # ログインホスト名を指定
User 利用者番号                      # 利用者番号を指定
IdentityFile ~/.ssh/id_rsa_cc       # 秘密鍵の保存場所とファイル名を指定

# フロントエンドサーバの設定（ホスト名を”front”とする場合）
Host front
HostName front.cc.tohoku.ac.jp
User 利用者番号
ProxyCommand ssh -CW %h:%p login    # login 経由で多段 SSH する設定
IdentityFile ~/.ssh/id_rsa_cc
```

○データ転送サーバを利用するための設定

```
# データ転送サーバの設定（ホスト名を”file”とする場合）
Host file
HostName file.cc.tohoku.ac.jp
User 利用者番号
IdentityFile ~/.ssh/id_rsa_cc
```

○HPCI 用ログインノードを利用するための設定

```
# HPCI 用ログインノードの設定（ホスト名を”hpcif”とする場合）
Host hpcif
HostName hpcif.cc.tohoku.ac.jp
User 利用者番号
IdentityFile ~/.ssh/id_rsa_cc
```

（※）Windows の場合、フロントエンドサーバへのログイン時に以下のようなエラーが出る場合があります。

```
$ ssh front
CreateProcessW failed error:2
posix_spawn: No such file or directory
```

エラーが出た場合は次の要領で config ファイルを書き換えてください。

[1] ターミナルソフトを起動し、以下のコマンドで ssh の絶対パスを調べる。

```
$ gcm ssh
CommandType   Name    Version   Source
-----------   ----    -------   ------
Application   ssh.exe x.x.x     C:¥WINDOWS¥System32¥OpenSSH¥ssh.exe
```

[2] config ファイルの「ProxyCommand ssh …」の行の「ssh」の部分を、絶対パス（[1]で「Source」に表示された文字列）に書き換える。

```
# 修正前
ProxyCommand ssh -CW %h:%p login
# 修正後
ProxyCommand C:¥WINDOWS¥System32¥OpenSSH¥ssh.exe -CW %h:%p login
```

5.2. フロントエンドサーバのログイン方法

ターミナルソフトを起動し、以下のコマンドを実行するとログインします。ホスト名を別の文字列で設定している場合は「front」の部分を読み替えてください。

```
$ ssh front
```

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 front1 または front2 が選択されます。どちらにログインしても、動作は変わりません。

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。

5.3. データ転送サーバの利用方法

データ転送サーバは、ログインして利用するのではなく、利用者のローカル PC 上から scp コマンドや sftp コマンドで利用します。詳しくは以下をご参照ください。

データ転送（ストレージ）: https://www.ss.cc.tohoku.ac.jp/storage/

5.4. HPCI 用ログインノードのログイン方法

ターミナルソフトを起動し、以下のコマンドを実行するとログインします。ホスト名を別の文字列で設定した場合は「hpcif」の部分を読み替えてください。

```
$ ssh hpcif
```

5.5. ログインシェルの確認と変更

ログインシェルは、デフォルトでは csh が設定されています。設定の確認および変更は以下の手順で行います。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。

(1) フロントエンドサーバにログインする。

(2) 以下のコマンドを実行する。

○ログインシェルの確認

```
front1 $ fchsh　（ログインシェルの確認）
Enter Password:　（パスワードを入力）
loginShell: /bin/tcsh　（現在のログインシェルが表示される）
```

○ログインシェルの変更

```
front1 $ fchsh /bin/bash　（ログインシェルを/bin/bash に変更）
Enter Password:　（パスワードを入力）
Changed loginShell to /bin/bash　（ログインシェルが変更された）
```

6. パスワードの変更

利用者ポータルなどで使用するパスワードの変更は、以下の手順で行います。

(1) 以下の URL 先から利用者ポータルを開きます。

利用者ポータルには、利用者番号と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パスワード
- 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のパスワード

7. おわりに

本稿では、鍵ペアの作成とログイン方法についてご紹介しました。センターのシステムを安全にご利用いただければ幸いです。ご不明な点、ご質問等ございましたら、お気軽にセンター（利用相談）までお問い合わせください。

利用相談：https://www.ss.cc.tohoku.ac.jp/consultation/

また、センターからのお知らせは、ウェブサイトにてご確認ください。

センターウェブサイト：https://www.ss.cc.tohoku.ac.jp/

［大規模科学計算システム］

サブシステム AOBA-A の利用法

情報部情報基盤課 共同利用支援係 共同研究支援係
サイバーサイエンスセンター スーパーコンピューティング研究部

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-A の運用を 2020 年 10 月から開始しています。本稿では，サブシステム AOBA-A でのプログラミング利用ガイドとして，プログラムの作成からコンパイル，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「利用申請」(https://www.ss.cc.tohoku.ac.jp/apply-for-use/) ご参照ください。

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-A（SX-Aurora TSUBASA）とサブシステム AOBA-B（LX 406Rz-2）の 2 つの計算機システムをサービスしています（図 1）。また，2PB のストレージシステムは AOBA-A，AOBA-B と InfiniBand で接続され，高速な I/O が可能です。

利用者は SINET5 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。

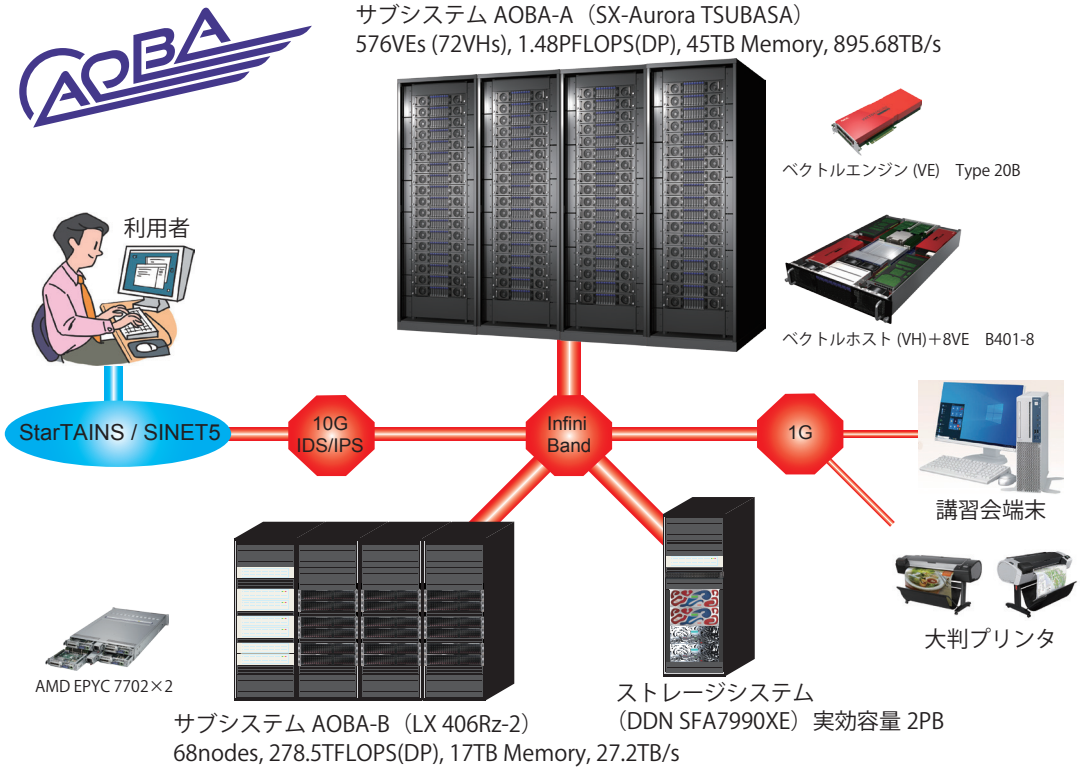


図 1　スーパーコンピュータ AOBA の構成

2.1　サブシステム AOBA-A の特徴

2.1.1　SX-Aurora TSUBASA アーキテクチャ

SX-Aurora TSUBASA（日本電気株式会社製）は，前システム SX-ACE 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，及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 2,456GFLOPS(DP) となるマルチコア (8 コア) ベクトルプロセッサを 1 基，主記憶は 48GB を搭載し，1.53TB/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 SX-Aurora TSUBASA のシステムは，1VH と 8VE が構成単位となる B401-8 モデルを採用し，システム全体では 72 個の VH と 576 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，1.48PFLOPS(DP)，主記憶は 45TB，総メモリバンド幅は 895.68TB/s となります。

図 2　サブシステム AOBA-A（4 ラック）

- マルチコアベクトルエンジン（8 コア）あたり，2,456GFLOPS(DP) の理論演算性能
- 1VE あたり 48GB 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 1.53TB/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。

サブシステム AOBA-A 向けにインストールされているアプリケーションについては，「アプリケーションサービス」(https://www.ss.cc.tohoku.ac.jp/software-service/) をご参照ください。

■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++ コンパイラが利用出来ます。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 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を利用出来ます。NEC Numeric Library Collection（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 に対応しています。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 種類があります。

■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 8 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 48GB まで利用出来ます。

■自動並列化 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，プログラムをそのまま利用することができます。

共有メモリ並列実行が可能なので，VE 内の 8 コアまで，メモリは 48GB まで利用出来ます。

■OpenMP 並列 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。

自動並列化と同じく，共有メモリ並列実行が可能なので，VE 内の 8 コアまで，メモリは 48GB まで利用出来ます。

■MPI 並列 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。センターでは最大 256VE（2,048 コア），メモリは 12TB まで利用出来ます。

MPI 並列と自動並列化／ 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	-	-	8 コア	48GB
自動並列化	コンパイラによる自動並列化	なし	8 コア	48GB
OpenMP 並列	ユーザによる指示行挿入	少ない	8 コア	48GB
MPI 並列	MPI ライブラリを用いたプログラミング	多い	2,048 コア	12,288GB

3　プログラムのコンパイルと実行手順

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

3.1　フロントエンドサーバへのログイン

サブシステム AOBA-A 用のコンパイルはフロントエンドサーバで行います。セキュリティ対策のため，フロントエンドサーバへはログインサーバを経由します。ストレージシステムとの大容量のデータ転送は，データ転送サーバを利用します。いずれのサーバにも公開鍵暗号方式による SSH 接続でログインを行います。

図 3　ログインサーバ，フロントエンドサーバ，データ転送サーバ

鍵の作成からログインサーバを経由してフロントエンドサーバーへログインする方法については，利用申請からログインまで (https://www.ss.cc.tohoku.ac.jp/first-use/) をご参照ください。

データ転送サーバの利用方法や，ストレージ容量の追加方法などについては，「データ転送（ストレージ）」(https://www.ss.cc.tohoku.ac.jp/storage/) をご参照ください。

3.2　プログラムの作成とコンパイル

3.2.1　ソースファイルの作成

フロントエンドサーバ上でソースファイルを作成するためのエディタは，emacs または vim が一般的です。研究室等のサーバやパソコンからソースファイルを転送する場合は，SSH 対応の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 用にするためには，転送した後のソースコードに nkf コマンドを利用します（リスト 1）。

リスト 1　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
front$ nkf -Lu 変換前ソースファイル名 > 変換後ソースファイル名

改行コードをLF，文字コードをUTF-8に変換してソースファイルを上書き保存する方法
front$ nkf --overwrite -Lu ソースファイル名
```

3.2.2　コンパイラの仕様

SX-Aurora TSUBASA 用に最適化された Fortran 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
- 自動ベクトル化・自動並列化機能機能対応
- ISO/IEC 1539-1:2004 Programming languages - Fortran に準拠
- OpenMP Application Program Interface Version 4.5 に準拠
- ISO/IEC 1539-1:2010 Programming languages - Fortran 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
- 自動ベクトル化・自動並列化機能機能対応
- ISO/IEC 9899:2011 Programming languages - C に準拠
- ISO/IEC 14882:2014 Programming languages - C++ に準拠
- OpenMP Application Program Interface Version 4.5 に準拠

3.2.3　コンパイルを行う

ソースファイルはそれぞれの言語用コマンドでコンパイルします。単一コアで実行する逐次実行，自動並列化または OpenMP 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　ソースコードは nfort コマンドまたは mpinfort コマンドでコンパイルします。利用したい機能があれば，表 2 に示したようなコンパイルオプションを同時に指定します。その他のオプションについてはマニュアル（5 章）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，カレントディレクトリに実行オブジェクト a.out が作成されます。

表 2　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 （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ありますのでご注意下さい）
-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 （-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出力する
-fbounds-check	バウンスチェック（配列の上下限のチェック）を行う

リスト 2　nfort コマンドの使用例

```
（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

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

リスト 3　mpinfort コマンドの使用例

```
（MPI並列実行）
front$ 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）
front$ 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）
front$ 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。利用したい機能があれば Fortran コンパイラと同じく，表 2 に示したようなコンパイルオプションを同時に指定します。その他のオプションについてはマニュアル（5 章）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，カレントディレクトリに実行オブジェクト a.out が作成されます。

リスト 4　ncc/nc++ コマンドの使用例

```
（逐次実行）
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

（自動並列化実行）
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並列実行）
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
```

リスト 5　mpincc/mpinc++ コマンドの使用例

```
（MPI並列実行）
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

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

■性能解析情報（FTRACE）を確認する　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。

結果の確認は，ftrace コマンドでテキスト形式で確認するか，ftraceviewer コマンドで GUI で確認することが出来ます。図 4 は Ftrace Viewer の表示例です。

FTRACE と Ftrace Viewer についての詳細は，5 章に記載のマニュアルをご参照ください。

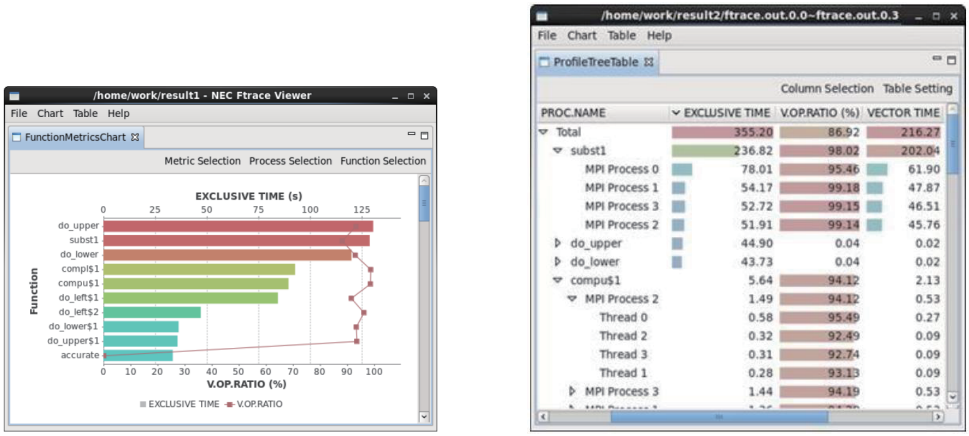


図 4　Ftrace Viewer の表示例

3.3　プログラムの実行

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（ジョブ）の実行は，バッチリクエスト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リクエストの処理に NEC Network Queuing System V（以下，NQSV）を採用しています。図 5 にバッチリクエストの概念図を示します。

詳しくは「ジョブの実行方法」(https://www.ss.cc.tohoku.ac.jp/nqs/) をご参照ください。

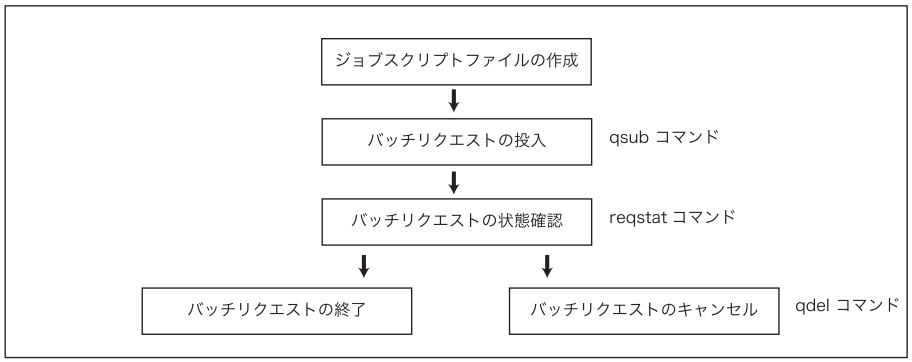


図 5　バッチリクエストの概念図

3.3.1　ジョブスクリプト

ジョブスクリプトに指定が必要となるのは,

- 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

です。他に環境変数の指定，ファイルの操作コマンド等があれば適切な箇所に手続きを記述します。オプション，環境変数は ジョブスクリプト中では#PBS の後に記述します。

サブシステム AOBA-A で実行する場合の，必須オプションを表 3 に，利用形態を表 4 に示します。

表 3　サブシステム AOBA-A で実行する場合の必須オプション

オプション	指定するもの	機能
-q	sx sxmix sxf のいずれか 1 つ	AOBA-A の利用形態を指定
--venode	1〜256 の整数値	利用する VE 数を指定
-l elapstim_req	最大経過時間を hh:mm:ss 形式で	計算機を利用する時間を指定

表 4　サブシステム AOBA-A で実行する場合の利用形態

投入キュー名	利用可能 VE 数	リクエストの実行形態
sxf	1	無料の 1VE リクエスト（VH を共用する）
sx	1	1VE リクエスト（VH を共用する）
sx	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）

VH を共用しない：　投入したリクエストが他のリクエストと VH を共用しないで実行されます。2〜7 個の VE を使うと指定をしたリクエストも，8 個の VE と 1 個の VH を確保します。他のリクエストと VH を共用しないため，演算時間のバラツキが少なくなります。

VH を共用する： 投入したリクエストは他のリクエストと VH を共用して実行されます。2 個の VE を使うと指定したリクエストは，他 6 個の VE で別のリクエストが実行されることがあります。指定した数の VE（2〜8 個）を確保しやすく，混雑時にも待ち時間が短くなります。

3.3.2 ジョブスクリプトの例

■逐次実行の場合 リスト 6 とリスト 7 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 中の 1 コアで実行されます。

リスト 6 ジョブスクリプトの例（逐次実行）

```
（run.csh の記述内容）
#!/bin/csh
#PBS -q sx #AOBA-A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 7 ジョブスクリプトの例（逐次実行，無料キュー）

```
（run.csh の記述内容）
#!/bin/csh
#PBS -q sxf #AOBA-A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 8 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 2〜8 コアで実行されます。実行コア数の指定は，環境変数 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 8 ジョブスクリプトの例（自動並列化/OpenMP 並列実行）

```
（run.csh の記述内容）
#!/bin/csh
#PBS -q sx #AOBA-A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 OMP_NUM_THREADS=8 #8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 9 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。（逐次実行や自動並列化/OpenMP 並列実行用にコンパイルされたプログラムは，MPI 並列実行を行っても 1VE 中の 8 コア並列でしか実行されません。）

MPI プロセス数が確保した VE の物理コア数（VE 数× 8）を超えると，演算性能が著しく低下しますのでご注意下さい。

リスト 9 ジョブスクリプトの例（MPI 並列実行）

```
（run.csh の記述内容）
#!/bin/csh
#PBS -q sx #AOBA-A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 64 ./a.out #カレントディレクトリの a.out を64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 10 に MPI と自動並列化／ OpenMP 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× 8）を超えると，演算性能が著しく低下しますのでご注意下さい。

リスト 10　ジョブスクリプトの例（MPI と自動並列化／ OpenMP 同時並列実行）

```
（run.csh の記述内容）
#!/bin/csh
#PBS -q sx #AOBA-A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 OMP_NUM_THREADS=8 #VE内は8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×8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：　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 11 の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します。

リスト 11　出力をプロセス毎に分割する方法

```
（run.csh の記述内容）
#!/bin/csh
#PBS -q sx #AOBA-A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 #出力形式に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 64 /opt/nec/ve/bin/mpisep.sh ./a.out #カレントディレクトリの a.out を64プロセス並列で実行
```

表 5　NMPI_SEPSELECT の引数と動作

NMPI_SEPSELECT の引数	動作
1	各 MPI プロセスの標準出力のみを stdout.uuu:rrr へ格納
2	各 MPI プロセスの標準エラーのみを stderr.uuu:rrr へ格納 (既定値)
3	各 MPI プロセスの標準出力と標準エラーをそれぞれ stdout.uuu:rrr および stderr.uuu:rrr へ格納
4	各 MPI プロセスの標準出力と標準エラーを同一ファイル stdout.uuu:rrr へ格納

実行時，stdout.*や stderr.*の同名ファイルが存在する場合は，上書きでなく追記します。

4　スーパーコンピュータ用数値演算ライブラリ

4.1　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。

4.1.1 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 6 に示すライブラリが使用できます。

表 6 Fortran 用 NLC ライブラリと機能概要

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 科学技術計算ライブラリ
ASL 統合インタフェース	フーリエ変換，乱数，ソート
ASL 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

4.1.2 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 7 に示すライブラリが使用できます。

表 7 C 言語用 NLC ライブラリと機能概要

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 科学技術計算ライブラリ
ASL 統合インタフェース	フーリエ変換，乱数，ソート
ASL 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
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 ライブラリの詳細およびコンパイルとリンク方法については 5 章の NLC (NEC Numeric Library Collection) ユーザーズガイドをご参照ください。

5 マニュアル

サブシステム AOBA-A についてのマニュアルは，NEC Aurora Forum の NEC SX-Aurora TSUBASA Documentation をご参照下さい。英語版，日本語版ともに提供されています。

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。

5.1　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下さい。

- 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 MPI ユーザーズガイド, NEC MPI User's Guide

5.2　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下さい。

- 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

5.3　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下さい。

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

6　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-A 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，ご質問等ございましたら，お気軽にセンターまでお問い合わせください。問い合わせについては「利用相談」(https://www.ss.cc.tohoku.ac.jp/consultation/) をご参照下さい。

［大規模科学計算システム］

サブシステム AOBA-B の利用法

情報部情報基盤課 共同利用支援係 共同研究支援係
サイバーサイエンスセンター スーパーコンピューティング研究部

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-B の運用を 2020 年 10 月から開始しています。本稿では，サブシステム AOBA-B でのプログラミング利用ガイドとして，プログラムの作成からコンパイル，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「利用申請」(https://www.ss.cc.tohoku.ac.jp/apply-for-use/) ご参照ください。

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-A（SX-Aurora TSUBASA）とサブシステム AOBA-B（LX 406Rz-2）の 2 つの計算機システムをサービスしています（図 1）。また，2PB のストレージシステムは AOBA-A，AOBA-B と InfiniBand で接続され，高速な I/O が可能です。

利用者は SINET5 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。

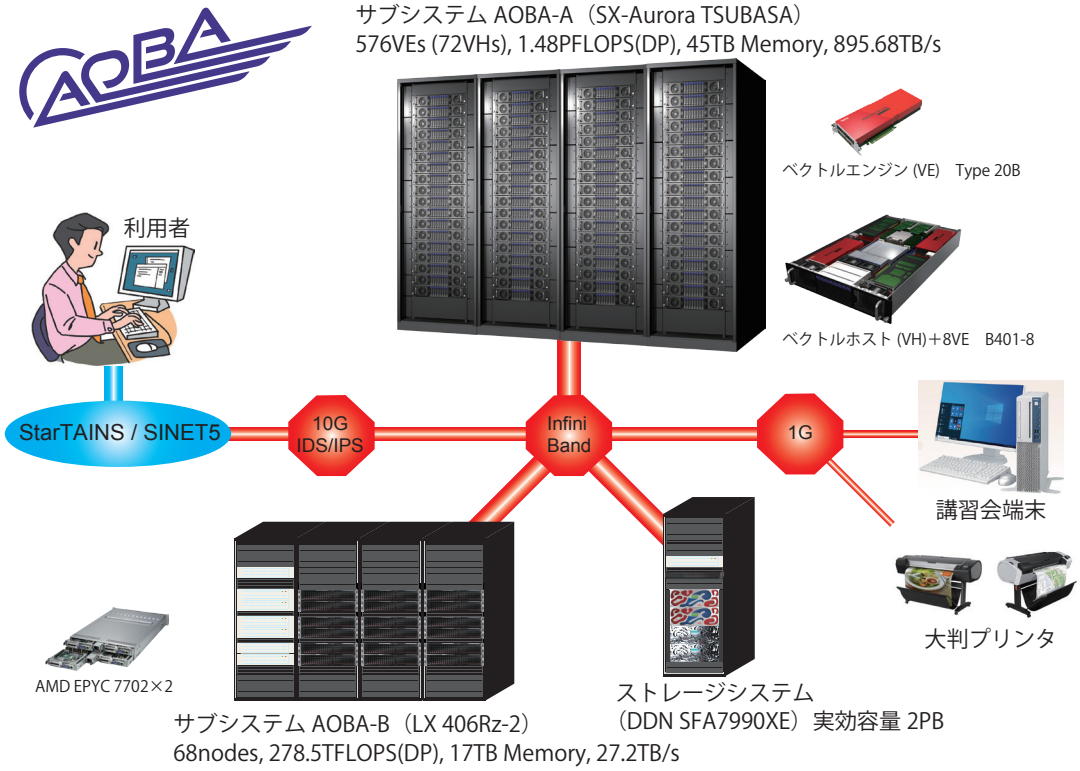


図 1　スーパーコンピュータ AOBA の構成

2.1　サブシステム AOBA-B の特徴

LX 406Rz-2 は，1 ノードに AMD EPYC プロセッサ 7702（64 コア）を 2 基と 256GB の主記憶装置を搭載し，合計 68 ノードで構成されます。OpenMP・MPI を利用したノード内の並列処理は 128 並列まで可能で，ノードあたりの最大演算性能は 4.096TFLOPS（倍精度），システム全体では 278.5 TFLOPS（倍精度）となります。複数のノードを使用した並列処理は，MPI の利用により最大 2,048 コア並列，4TB の主記憶を利用可能です。ベクトル演算に不向きなプログラムや，商用アプリケーションの高速な実行が可能です。

図 2　サブシステム AOBA-B（4 ラック）

- 1 ノード（64 コア× 2）あたり，4.096TFLOPS(DP) の理論演算性能
- 1 ノードあたり 256GB 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 409.6GB/s
- 一時領域として，アクセスが高速な SSD ストレージを搭載

2.1.1　ソフトウェア

フロントエンドサーバの Linux OS 環境上で，EPYC プロセッサ向けのプログラムを作成できます。一般的な Linux OS ですので，商用アプリケーションやオープンソフトウェアも実行が可能です。

サブシステム AOBA-B 向けにインストールされているアプリケーションについては，「アプリケーションサービス」(https://www.ss.cc.tohoku.ac.jp/software-service/) をご参照ください。

■プログラミング言語　Fortran/C/C++ コンパイラとして，AMD Optimizing C/C++ Compiler と GNU Compiler Collection が利用出来ます。OpenMP による共有メモリ並列実行と MPI ライブラリによる分散メモリ並列実行が可能です。また，前システムの並列コンピュータで実行していたソースコードの移行用として，Intel Compiler をライセンス数限定で利用できます。

■科学技術計算ライブラリ　EPYC プロセッサに最適化された，AMD Optimizing CPU Libraries (AOCL) を利用出来ます。

- AOCL-Sparse
- BLIS
- FFTW (Fastest Fourier Transform in the West)
- AMD Math Library (LibM)
- libFLAME
- ScaLAPACK
- AMD Random Number Generator Library

AOCL の詳細については https://developer.amd.com/tools-and-sdks/をご参照ください。

また，Intel Compiler からは MKL ライブラリが利用出来ます。MKL についての詳細は https://www.xlsoft.com/jp/products/intel/perflib/mkl/index.html をご参照ください。

■アプリケーション性能解析ツール　AMD μ Prof が利用できます。実行したプログラムの性能およびシステム性能分析機能が利用出来ます。

AMD μ Prof の詳細については https://developer.amd.com/tools-and-sdks/をご参照ください。

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 3 種類が利用できます。並列実行には 3 種類があります。

■逐次実行　単一のコアで動作するプログラムです。1 ノード内には 64 コアの CPU が 2 つありますが，逐次実行では 1 コアのみが利用されます。メモリは 256GB まで利用出来ます。

■OpenMP 並列　逐次処理プログラムをユーザが明示的に並列化します。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。

共有メモリ並列実行が可能なので，1 ノード内の 128 コア（2CPU）を利用出来ます。メモリは 256GB まで利用出来ます。

■MPI 並列　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，複数のノードを用いた実行が可能です。サブシステム AOBA-B では最大 16 ノード（2,048 コア），メモリは 4,096GB まで利用出来ます。

MPI 並列と 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	-	-	128 コア	256GB
OpenMP 並列	ユーザによる指示行挿入	少ない	128 コア	256GB
MPI 並列	MPI ライブラリを用いたプログラミング	多い	2,048 コア	4,096GB

3　プログラムのコンパイルと実行手順

本章ではプログラムのコンパイルと，サブシステム AOBA-B で実行する手順を説明します。コンパイラについては AOCC を使用した場合について説明します。

3.1　フロントエンドサーバへのログイン

サブシステム AOBA-B 用のコンパイルはフロントエンドサーバで行います。セキュリティ対策のため，フロントエンドサーバへはログインサーバを経由します。ストレージシステムとの大容量のデータ転送は，データ転送サーバを利用します。いずれのサーバにも公開鍵暗号方式による SSH 接続でログインを行います。

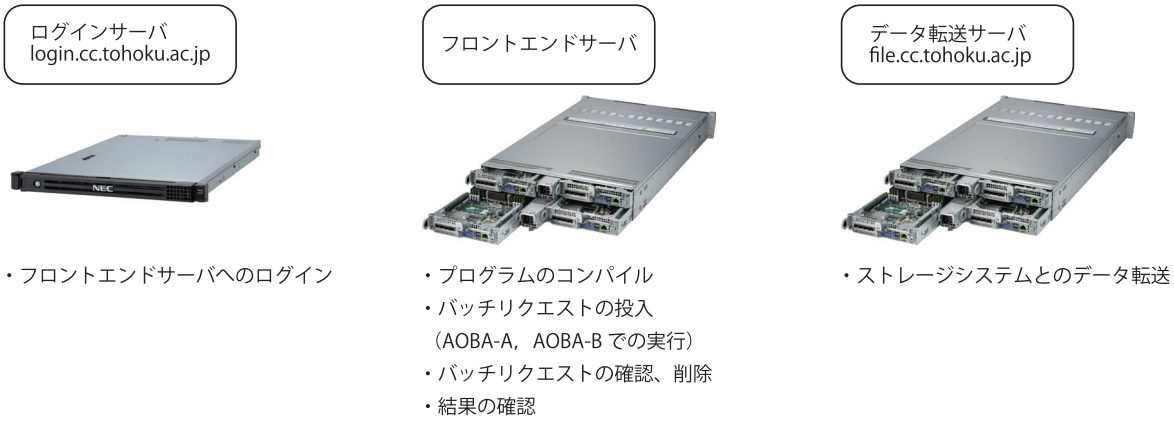


図 3　ログインサーバ，フロントエンドサーバ，データ転送サーバ

鍵の作成からログインサーバを経由してフロントエンドサーバーへログインする方法については，利用申請からログインまで (https://www.ss.cc.tohoku.ac.jp/first-use/) をご参照ください。

データ転送サーバの利用方法や，ストレージ容量の追加方法などについては，「データ転送（ストレージ）」(https://www.ss.cc.tohoku.ac.jp/storage/) をご参照ください。

3.2　プログラムの作成とコンパイル

3.2.1　ソースファイルの作成

フロントエンドサーバ上でソースファイルを作成するためのエディタは，emacs または vim が一般的です。研究室等のサーバやパソコンからソースファイルを転送する場合は，SSH 対応の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。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 用にするためには，転送した後のソースコードに nkf コマンドを利用します（リスト 1）。

リスト 1　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
front$ nkf -Lu 変換前ソースファイル名 > 変換後ソースファイル名

改行コードをLF，文字コードをUTF-8に変換してソースファイルを上書き保存する方法
front$ nkf --overwrite -Lu ソースファイル名
```

3.2.2　利用可能なコンパイラの種類

AOCC コンパイラとして，以下の Fortran および C/C++ コンパイラを利用できます。

AOCC

- flang：Fortran コンパイラ
- mpifort：MPI プログラム用 Fortran コンパイラ
- clang：C コンパイラ
- mpicc：MPI プログラム用 C コンパイラ
- clang++：C++ コンパイラ
- mpic++：MPI プログラム用 C++ コンパイラ

3.2.3 コンパイルを行う

ソースファイルはそれぞれの言語用コマンドでコンパイルします。単一コアで実行する逐次実行，OpenMP 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 ソースコードは flang コマンドまたは mpifort コマンドでコンパイルします。利用したい機能があれば，表 2 に示したようなコンパイルオプションを同時に指定します。その他のオプションについてはマニュアル（4 章）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 cpp 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，標準ではカレントディレクトリに実行オブジェクト a.out が作成されます。

表 2 Fortran,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
-march=znver2	EPYC CPU（Rome）用にコンパイルすることを指定する
-On	最適化レベルを指定する
n に指定できる値	
fast	最大限の最適化を適用する
3	積極的に最適化を適用する
2	（既定値）標準的な最適化を適用する
1	最低限の最適化を適用する
0	すべての最適化を無効化する
-fopenmp	OpenMP 並列化を利用する

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ありますのでご注意下さい。

リスト 2 flang コマンドの使用例

```
（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

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

リスト 3 mpifort コマンドの使用例

```
（MPI並列実行）
front$ mpifort 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）
front$ mpifort -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 clang または mpicc コマンドで C プログラムを，clang++ または mpic++ コマンドで C++ プログラムをコンパイルします。利用したい機能があれば Fortran コンパイラと同じく，表 2 に示したようなコンパイルオプションを同時に指定します。その他のオプションについてはマニュアル（4 章）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，カレントディレクトリに実行オブジェクト a.out が作成されます。

リスト 4　clang/clang++ コマンドの使用例

```
（逐次実行）
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並列実行）
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
```

リスト 5　mpicc/mpic++ コマンドの使用例

```
（MPI並列実行）
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

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

3.3　プログラムの実行

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（ジョブ）の実行は，バッチリクエスト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リクエストの処理に NEC Network Queuing System V（以下，NQSV）を採用しています。図 4 にバッチリクエストの概念図を示します。

詳しくは「ジョブの実行方法」(https://www.ss.cc.tohoku.ac.jp/nqs/) をご参照ください。

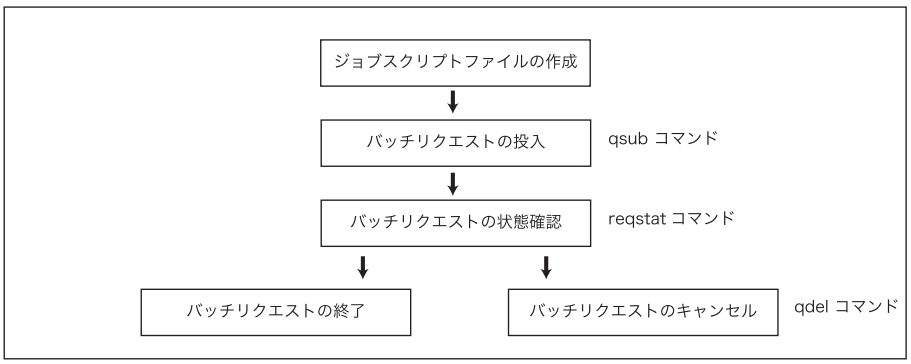


図 4　バッチリクエストの概念図

3.3.1　ジョブスクリプト

ジョブスクリプトに指定が必要となるのは，

- 投入キュー名
- 利用するノード数
- 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

です。他に環境変数の指定，ファイルの操作コマンド等があれば適切な箇所に手続きを記述します。オプション，環境変数は ジョブスクリプト中では#PBS の後に記述します。

サブシステム AOBA-B で実行する場合の，必須オプションを表 3 に示します。

表 3　サブシステム AOBA-B で実行する場合の必須オプション

オプション	指定するもの	機能
-q	lx	AOBA-B を利用することを指定
-b	1〜16 の整数値	利用するノード数を指定
-l elapstim_req	最大経過時間を hh:mm:ss 形式で	計算機を利用する時間を指定

3.3.2　ジョブスクリプトの例

■逐次実行の場合　リスト 6 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 1CPU 中の 1 コアで実行されます。

リスト 6　ジョブスクリプトの例（逐次実行）

```
（run.csh の記述内容）
#!/bin/csh
#PBS -q lx #AOBA-Bを使用する
#PBS -b 1 #1ノードを使う
#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
```

■OpenMP 並列実行の場合　リスト 7 に OpenMP 並列実行の場合のジョブスクリプトの例を示します。OpenMP 並列実行の場合，プログラムは 1 ノード内の 2〜128 コアで実行されます。実行コア数の指定は，環境変数 OMP_NUM_THREADS で行います。

リスト 7　ジョブスクリプトの例（OpenMP 並列実行）

```
（run.csh の記述内容）
#!/bin/csh
#PBS -q lx #AOBA-Bを使用する
#PBS -b 1 #1ノードを使う
#PBS -l elapstim_req=2:00:00 #最大経過時間を2時間に指定
#PBS -v OMP_NUM_THREADS=128 #128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 8 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。mpirun コマンドの後に，-b オプションで指定した使用ノード数に応じて，バッチリクエストのノード割当てを自動的に設定するための環境変数$NQSV_MPIOPT を指定します。（逐次実行や OpenMP 並列実行用にコンパイルされたプログラムは MPI 並列実行を行っても，1 ノード内でしか実行されません。）

MPI プロセス数が確保したノードの物理コア数（ノード数× 128）を超えると，演算性能が著しく低下しますのでご注意下さい。

リスト 8　ジョブスクリプトの例（MPI 並列実行）

```
（run.csh の記述内容）
#!/bin/csh
#PBS -T openmpi # MPI実行環境を指定，AOCCではopenmpiを使用する
#PBS -q lx #AOBA-Bを使用する
#PBS -b 1 #1ノードを使う
#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 $NQSV_MPIOPTS ./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，OpenMP 並列実行同時利用の場合　リスト 9 に MPI，OpenMP 並列実行同時利用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）×（ノード内並列数）が確保したノードの物理コア数（ノード数× 128）を超えると，演算性能が著しく低下しますのでご注意下さい。

リスト 9　ジョブスクリプトの例（MPI，OpenMP 同時並列実行）

```
（run.csh の記述内容）
#!/bin/csh
#PBS -T openmpi # MPI実行環境を指定，AOCCではopenmpiを使用する
#PBS -q lx #AOBA-Bを使用する
#PBS -b 2 #2ノードを使う
#PBS -l elapstim_req=2:00:00 #最大経過時間を2時間に指定
#PBS -v OMP_NUM_THREADS=64 #ノード内は64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 4 $NQSV_MPIOPTS ./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
```

4　マニュアル

4.1　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下さい。

- AMD Optimizing C/C++ Compiler：https://developer.amd.com/amd-aocc/
- Intel Compiler：https://www.xlsoft.com/jp/products/intel/studio_xe/

4.2　数値計算ライブラリマニュアル

並列コンピュータ用数値計算ライブラリについては以下のマニュアルをご参照下さい。

- AMD Optimizing CPU Libraries（AOCL）：https://developer.amd.com/amd-aocl/
- Intel MKL：https://www.xlsoft.com/jp/products/intel/perflib/mkl/

5　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-B 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，ご質問等ございましたら，お気軽にセンターまでお問い合わせください。問い合わせについては「利用相談」(https://www.ss.cc.tohoku.ac.jp/consultation/) をご参照下さい。

［大規模科学計算システム］

バッチリクエストによるジョブの実行方法

情報部情報基盤課 共同研究支援係 共同利用支援係

1. はじめに

本センターの計算機は、バッチリクエストの方法で使用します。ジョブの実行をバッチ処理で行う方法です。ジョブ管理システムは NEC Network Queuing System V（以下、NQSV)を採用しています。

本稿では、バッチリクエストによるジョブの実行方法についてご紹介します。

2. 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、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。実行すべきジョブをシェルスクリプト（ジョブスクリプト）に記述し、NQSV に投入することにより実現されます。

図 1 に、バッチリクエストの実行の流れを示します。①は 4 章、②は 5 章、③は 6〜8 章、④は 9 章で詳しく解説します。

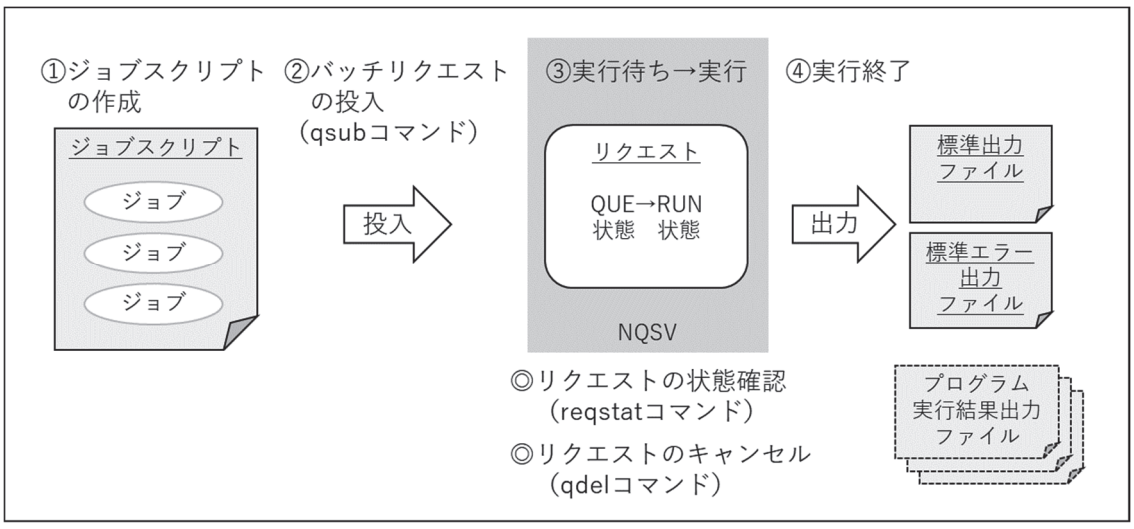


図 1　バッチリクエストの実行の流れ

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。利用する計算資源（投入キュー名、利用 VE 数／ノード数、最大経過時間）等も記述します。

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を投入する
reqstat コマンド　投入したリクエストの状態を表示する（状態確認）
qdel コマンド　　投入したリクエストをキャンセルする

④実行終了

リクエストの実行が終了すると標準出力ファイルと標準エラー出力ファイルが出力されます。終了したリクエストは reqstat コマンドの表示から消えます。

3. キュー構成

リクエストの投入先（キュー）について説明します。

表 1、表 2 に、新システムのキュー構成を示します。ジョブスクリプトで、投入キュー名、利用 VE 数／ノード数、最大経過時間を指定します。

表 1　サブシステム AOBA-A のキュー構成

利用形態	キュー名	VE 数	実行形態（※）	最大経過時間 既定値／最大値	メモリサイズ
無料	sxf	1	1VE	1 時間／1 時間	48GB×VE 数
共有	sx	1	1VE	72 時間／720 時間	
		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）		
占有	個別設定				

（VH：ベクトルホスト、VE：ベクトルエンジン）

（※）実行形態について

8VE 単位で確保 （VH を共用しない）	8VE 単位（1VH+8VE）で計算資源が確保され、投入したリクエストが他のリクエストと VH を共用しないで実行されます。他のリクエストと VH を共用しないため、演算時間のバラツキが少なくなります。
1VE 単位で確保 （VH を共用する）	1VE 単位で計算資源が確保され、投入したリクエストは他のリクエストと VH を共用して実行されます。指定した数の VE を確保しやすく、混雑時にも待ち時間が短くなります。
1VE	「VH を共用する」実行形態となります。

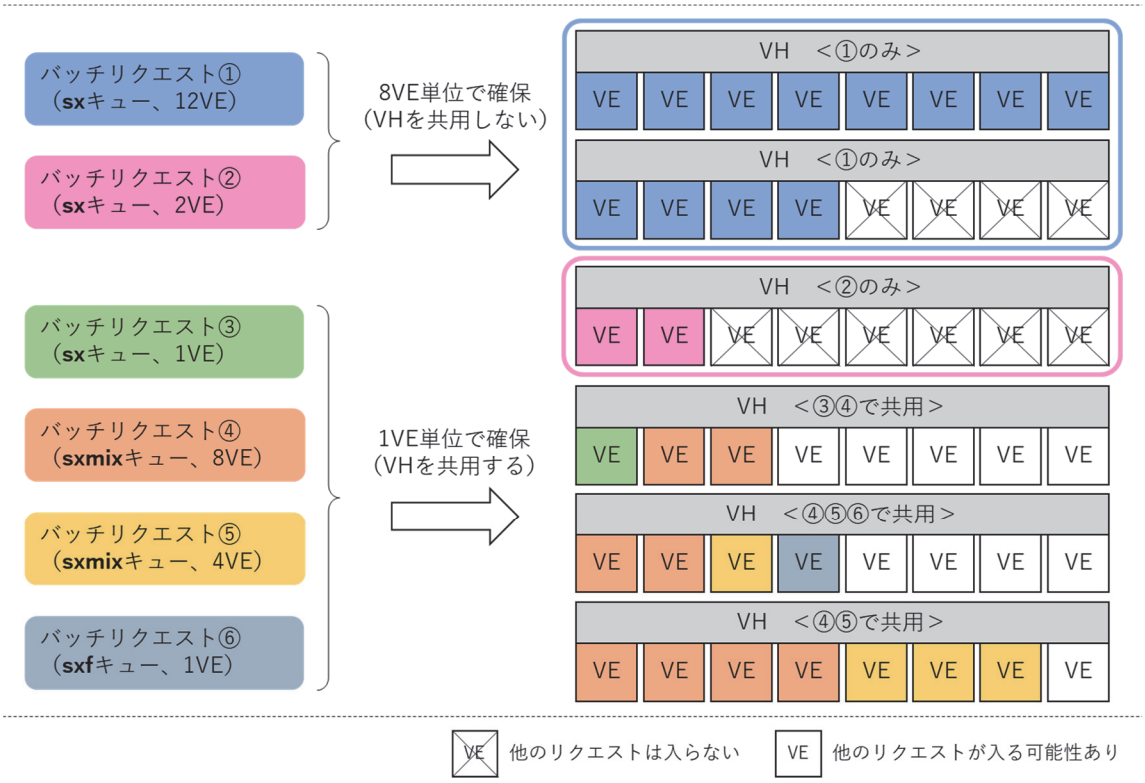


表 2　サブシステム AOBA-B のキュー構成

利用形態	キュー名	ノード数	最大経過時間 既定値／最大値	メモリサイズ
共有	lx	1～16	72 時間／720 時間	256GB×ノード数
占有	個別設定			

フロントエンドサーバ上での会話型での実行は、フロントエンドサーバに高い負荷がかかります。他の利用者への影響だけでなく、システム全体が正常に動作しなくなる恐れがあるため、行わないでください。負荷の高いプロセスは、管理者にて強制終了する場合があります。

4.　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じように、テキストファイルに、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 csh スクリプト、sh スクリプト、どちらも使用できます。ファイル名は任意の名前をつけることができます。（以降の解説は csh スクリプトで記述し、ファイル名は run.csh とします）

ジョブスクリプトに記述する基本項目は、以下のとおりです。リスト 1 に、ジョブスク

リプトの例を示します。

- 投入キュー名
- 利用 VE 数／ノード数
- 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

他に、環境変数の指定やファイルの操作コマンド等があれば、適切な箇所に記述します。

リスト 1　ジョブスクリプトの例

```
（run.csh の記述内容）
#!/bin/csh
#PBS -q sx                          # 投入キュー名
#PBS --venode 1                     # 利用 VE 数
#PBS -l elapstim_req=2:00:00  # 最大経過時間
cd $PBS_O_WORKDIR             # 作業ディレクトリへの移動コマンド
./a.out                             # プログラムの実行コマンド
```

投入時オプション（バッチリクエスト実行のための設定情報。qsub コマンドのオプションや環境変数の指定）は、最初のシェルコマンドが現れる前のコメント部分に、行の先頭に#PBS という文字列をつけて記述します。表 3 に、qsub コマンドの主なオプションを示します。

表 3　qsub コマンドの主なオプション（○印：指定必須）

AOBA-A	AOBA-B	オプション	機能
○	○	-q 投入キュー名	投入キュー名を指定(※1)
○	-	--venode 利用 VE 数	AOBA-A で実行する場合、利用 VE 数を指定(※1) (注意:このオプションは先頭のハイフンが 2 つ)
	○	-b 利用ノード数	AOBA-B で実行する場合、利用ノード数を指定(※1)
○	○	-l elapstim_req=時間	最大経過時間を指定 （秒数または時:分:秒（hh:mm:ss 形式）） （省略時：既定値）(※1)(※2)
		-A プロジェクトコード	課金先のプロジェクトコードを指定 （省略時：デフォルトのプロジェクトコード）

		-N リクエスト名	リクエスト名を指定 （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 （省略時：リクエスト投入時のディレクトリに「リクエスト名.o リクエスト ID」のファイル名で出力）
		-e ファイル名	標準エラー出力のファイル名を指定 （省略時：リクエスト投入時のディレクトリに「リクエスト名.e リクエスト ID」のファイル名で出力）
		-jo	標準エラー出力を標準出力と同じファイルに出力 （省略時：別々のファイルに出力）
		-m b	リクエストが実行開始したときにメール送信 （省略時：メール送信しない）
		-m e	リクエストが実行終了したときにメール送信 （省略時：メール送信しない）
		-M メールアドレス	メールの送信先を指定 （省略時：利用者番号@front.cc.tohoku.ac.jp）
		-r {y | n}	リクエストのリラン可否を指定 （※3） （y：許可、n：許可しない、省略時：y）

（※1） キュー名、利用可能な VE 数／ノード数、最大経過時間の既定値／最大値は、3 章の表 1、表 2 を参照してください。

（※2）経過時間とは、バッチリクエストが実行開始してから終了するまでの時間です。最大でどのぐ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、実行待ちの時間が短くなります。ただし、指定した最大経過時間を超えると、実行は打ち切りとなり強制終了します。I/O が高負荷の場合、経過時間が想定より長くなることがあります。必要十分な最大経過時間を指定してください。

（※3）リランとは、リクエストをはじめから実行しなおすことです。計算機のメンテナンスや障害発生時に、管理者にて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（たとえば、実時間で時間を計測している場合）は、あらかじめ該当オプションを指定して投入してください。

5. 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 2 に、qsub コマンドの実行例を示します。run.csh というジョブスクリプトを投入する例です。

リスト 2　qsub コマンドの実行例

```
front$ qsub run.csh
```

バッチリクエストが正常に投入されると、システムからリスト 3 のようなメッセージが返ってきます。この例は、リクエスト ID は 1234.job が採番され、投入先は sx キュー、課金先はプロジェクトコード un0000 が指定されたことを示しています。リクエスト ID は、このあと、バッチリクエストの状態確認やキャンセルの際に必要になります。

バッチリクエストの投入に失敗した場合は、エラーメッセージが返ってきます。ジョブスクリプトの記述などに誤りがないか確認してください。

リスト 3　qsub コマンド実行時のシステムメッセージ例

```
front$ qsub run.csh
プロジェクトコード：un0000 にリクエストを投入します
Request 1234.job submitted to queue：sx
```

6. 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ジューリングで制御されます。バックフィルスケジューリングでは、計算資源が確保できたリクエストから実行開始します。実行に必要な計算資源量（利用 VE 数またはノード数×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。

7. 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／実行中のリクエストの状態が表示されます。利用者自身のリクエストが全て表示されます。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 4 に reqstat コマンドの表示項目、表 5 にリクエストの主な状態を示します。

リクエストの実行開始予定時刻は、StartTime に表示されます。実行開始予定時刻は、他のリクエストの終了状況により随時更新されます。通常、更新前の時刻よりも遅くなることはありませんが、障害発生時などはその限りではありません。

表 4　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	課金先のプロジェクトコード
Que	実行キュー名
Node	投入時に指定した VE 数またはノード数
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（実行開始したら、実際に実行開始した時刻）
Memory	現在の使用メモリ量
ElapseTime	現在までの経過時間
NodeTime	現在までのノード時間（ElapseTime×Node）

表 5　リクエストの主な状態

表記	リクエストの状態
RUN	実行中
QUE	実行待ち
EXT	標準出力／標準エラー出力(リダイレクトも含む)の出力中（※）

（※） 以下のような場合、EXT 状態で長く停滞することがあります。他のリクエスト(他の利用者のリクエストも含む)の実行に影響がでる場合がありますので、十分ご留意ください。

- ファイル容量がクォータにかかり、標準出力／標準エラー出力を出力できない。
  空き容量が確保できるまで、EXT で停滞したままになります。ただちに、ファイルを整理して容量を空けてください。容量が確保できれば、正常に出力され、リクエストは終了します。なお、クォータにかからないよう、リクエスト投入前に、十分な空き容量があることを確認してください。なお、ファイル容量を追加することも可能です。詳しくは以下をご参照ください。
  データ転送(ストレージ) : https://www.ss.cc.tohoku.ac.jp/storage/

- システム上で問題が発生している。
  管理者にて対応しますので、そのままお待ちください。

8. 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したい場合は、qdel コマンドで行います。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。キャンセルは利用者自身で行うことができます。キャンセルできるリクエストは、利用者自身のリクエストのみです。

リスト 4 に、qdel コマンドの実行例を示します。1234.job というリクエストをキャンセルする例です。qdel に続けてキャンセルするリクエストのリクエスト ID を指定します。

リスト 4　qdel コマンドの実行例

```
front$ qdel 1234.job
```

システムがキャンセルを受け付けると、リスト 5 のようなメッセージを返します。

リスト 5　qdel コマンドの実行時のシステムメッセージ例

```
front$ qdel 1234.job
Request 1234.job was deleted.
```

9. 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。その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に出力された内容を格納した＜標準出力ファイル＞と、標準エラー出力に出力された内容を格納した＜標準エラー出力ファイル＞が作成されます。

投入時にオプション-o、-e を指定した場合はそのファイル名になります。指定しなかった場合は、以下のように命名されます。

<標準出力ファイル>　　リクエスト名.o リクエスト ID

<標準エラー出力ファイル>　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は 100MB 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、標準出力／標準エラー出力ではなく、ファイル名を指定して書き出すようにしてください。

10. マニュアル

各コマンドの詳しいオプションなどは、以下に公開されているマニュアル「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により、マニュアルに記載のとおりに動作しない場合もあります。

マニュアル：https://www.hpc.nec/documentation

11. おわりに

本稿では、バッチリクエストの投入と確認方法についてご紹介しました。センターの計

算資源を効率的にご活用いただければ幸いです。ご不明な点、ご質問等ございましたら、お気軽にセンター（利用相談）までお問い合わせください。

利用相談：https://www.ss.cc.tohoku.ac.jp/consultation/

［大規模科学計算システム］

ストレージシステムの利用法

情報部情報基盤課 共同研究支援係 共同利用支援係

1　はじめに

本稿では，スーパーコンピュータ AOBA のストレージシステムの利用法について紹介します．ストレージ環境にあるホームディレクトリと課題領域の容量確認方法と実行した結果ファイル等のデータ(ストレージ環境) をローカル PC へ転送する方法およびローカル PC からストレージ環境へ転送する方法について説明します．

2　ストレージ環境

2.1　ホームディレクトリ (uhome)

プログラムファイル等を置く自分専用のホームディレクトリになり，ScaTeFS マウントし，AOBA-A と AOBA-B の両方に共有しています．

ディレクトリ名： /uhome/利用者番号
クォータ（容量）制限： 5TB

クォータ制限を超過した場合，新規の書き込みができなくなりますのでご注意ください． クォータ制限を下回るように容量を削減すれば再度書き込みが可能になります．

- ホームディレクトリの容量確認コマンド

front$ uquota

表示例

Disk quotas for user 利用者番号

Filesystem	used(KB)	quota(TB)
/uhome/利用者番号	4	5

ホームディレクトリの容量追加申請については，2.3 章をご参照ください．

2.2　課題領域（/short/プロジェクトコード)

課題領域は事前申請となり，同一プロジェクトコードの利用者間で大規模なデータ容量を利用される領域になります．申請を希望される際は，ストレージ資源の兼ね合いもありますので，「利用相談」（https://www.ss.cc.tohoku.ac.jp/consultation/）から事前にご連絡をお願いします．

容量については，申請されたディスク容量 (quota 値)になります．ホームディレクトリ同様に AOBA-A と AOBA-B の両方に共有しています．

課題領域の容量については，以下のコマンドの quota(TB) 部分をご確認お願いします．

- 課題領域の容量確認コマンド
  front$ uquota -A プロジェクトコード
  表示例
  Disk quotas for project プロジェクトコード

Filesystem	used(KB)	quota(TB)
/short/プロジェクトコード	10	20

- **利用方法**
  - 同一プロジェクトコードの利用者間でデータを共有する．
  - 対象ディレクトリ内で利用者がそれぞれの専用サブディレクトリ (パーミッション：700) を作成し使用する．

利用方法の一例
プロジェクトコード：xx200001 の場合
コマンド例①）プロジェクトコードの利用者間で share を作成しデータを共有
front$ cp ホームディレクトリデータ /short/xx200001/share/

コマンド例②） 利用者の専用サブディレクトリを作成し使用
front$ mkdir /short/xx200001/利用者番号
front$ chmod 700 /short/xx200001/利用者番号
front$ cp ホームディレクトリデータ /short/xx200001/利用者番号/

【留意事項】
- ファイル同期コマンド（rsync コマンド），コピーコマンド (cp コマンド) を使用する際は，グループ権限を保持するオプションは設定せずにご利用ください．オプションを付けた場合，同一プロジェクトコード間のグループによる容量制限で正しく管理できなくなる恐れがあります．また，移動コマンド（mv コマンド）によるファイルの移動を行った場合，元のファイルのグループ権限が保持されてしまいますので，rsync コマンド，cp コマンドを利用するようにしてください．
- 課題領域を当年度までのご利用の際，翌年度はデータ保管を行っていません．猶予期間後，対象課題領域を削除しますので，ローカル PC のディスクへ移行を速やかに進めてください．

2.3 ストレージ申請

ファイル容量の追加は 1TB 単位から申請可能です．ホームディレクトリ，課題領域ともに利用負担金が発生しますので，詳しくは「利用負担金」(https://www.ss.cc.tohoku.ac.jp/charge/) をご参照ください．

申請用紙は，「ストレージ容量申請書」(https://www.ss.cc.tohoku.ac.jp/application-form/) を使い申請をお願いします．

3　データ転送方法

データ転送サーバへログインし，SSH による暗号化を行う scp(Secure CoPy)，SFTP(Ssh File Transfer Protocol) を利用します．接続方法については「利用申請からログインまで」(https://www.ss.cc.tohoku.ac.jp/first-use/) をご参照ください．

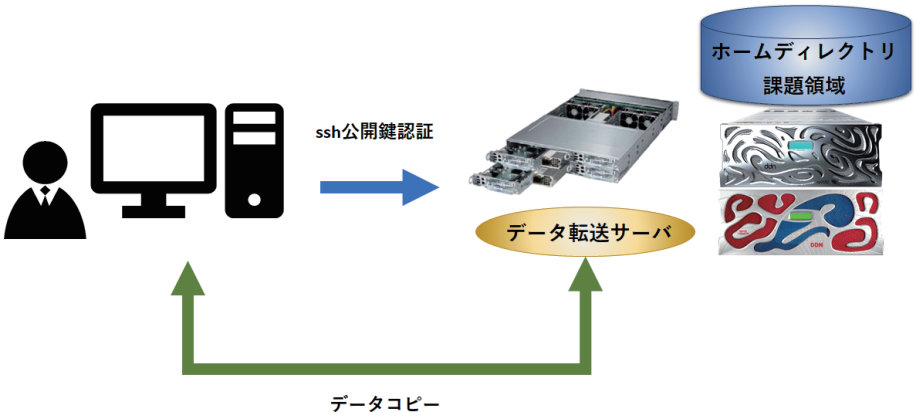


図 1　アクセスイメージ

3.1　Powershell(Windows)・MAC・Linux

標準で SSH クライアントがインストールされています．インストールされている各 Terminal ソフトでデータ転送サーバへログインします．

- **scp コマンド**

  SSH 利用し，ネットワーク・ホスト間でファイルを安全にコピーするためのコマンドです．

  ローカル PC からリモート (ストレージ環境) に転送

  $ scp ローカル PC の保存先パス　利用者番号@file.cc.tohoku.ac.jp:保存先パス

  scp コマンドの実行例

  (ローカル PC 上にある sample.txt ファイルをホームディレクトリへ転送)

  　　$ scp sample.txt 利用者番号@file.cc.tohoku.ac.jp:sample.txt

  パスフレーズを聞かれますので入力します．

  リモート (ストレージ環境) からローカル PC に転送

  $ scp 利用者番号@file.cc.tohoku.ac.jp:ファイル名 ローカル PC の保存先パス

  scp コマンドの実行例

  (ホームディレクトリ上にある sample.txt をローカル PC へ転送)

  　　$ scp 利用者番号@file.cc.tohoku.ac.jp:sample.txt ./

  詳しい用例については man コマンドを利用し，scp コマンドのマニュアル閲覧をお願いします．

  　　$ man scp

- **sftp コマンド**
  SSH 利用し，対話的なファイル転送を行うことができるコマンドです．

  ローカル PC からリモート (ストレージ環境) に転送
  $ sftp 利用者番号@file.cc.tohoku.ac.jp
  パスフレーズを聞かれますので入力します．
  sftp> と表示されたら成功です．
  sftp> put ファイル名 保存先フォルダ

  sftp コマンドの実行例
  (ローカル PC 上にある sample.txt ファイルをホームディレクトリへ転送)
  sftp> put パス名/sample.txt ./

  リモート (ストレージ環境) からローカル PC に転送
  $ sftp 利用者番号@file.cc.tohoku.ac.jp
  パスフレーズを聞かれますので入力します．
  sftp> と表示されたら成功です．
  sftp> get ファイル名 保存先フォルダ

  sftp コマンドの実行例
  (ホームディレクトリ上にある sample.txt をローカル PC へ転送)
  sftp> get ./sample.txt ./

  詳しい用例については man コマンドを利用し，sftp コマンドのマニュアル閲覧をお願いします．
  $ man sftp

3.2 WinSCP(Windows ソフト)

標準で scp，sftp に対応したソフトウェアがインストールされていないため，はじめにインストールする必要があります．
ここでは，代表的なソフトウェアである WinSCP を利用したファイル転送方法を説明します．

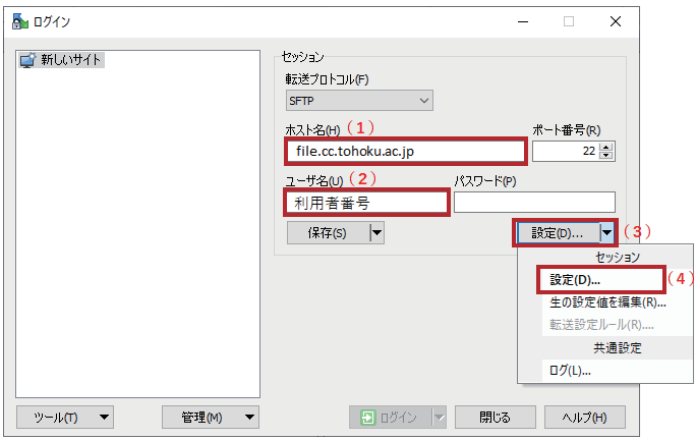


図 2　WinSCP 設定画面

1. WinSCP を起動します.
2. ホスト名 （上図（1））に file.cc.tohoku.ac.jp と入力します.
3. ユーザ名 （上図（2））に利用者番号を入力します.
4. 設定 （上図（3））のプルダウンメニューから設定 （上図（4））をクリックします.

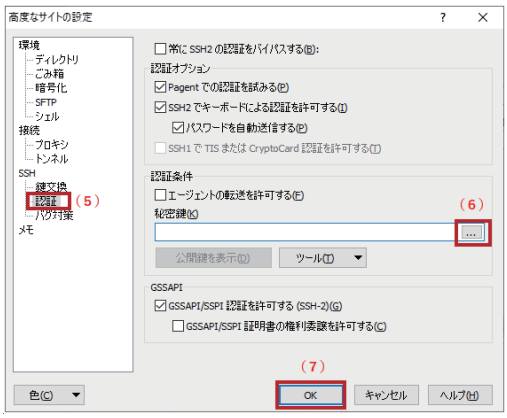


図 3　WinSCP 鍵認証設定画面

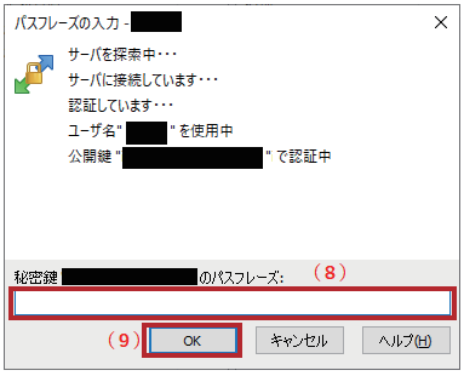


図 4　WinSCP 鍵認証画面

5. 左側ナビゲーションメニューの認証（上図（5））を選択します.
6. 秘密鍵のプルダウンメニュー （上図（6））をクリックし，ログインに使用する秘密鍵を指定します．秘密鍵 (.ppk ファイル) を未生成の場合，3.2.1 章をご参照ください.
7. OK （上図（7））をクリックし，鍵の設定を保存します.
8. 元の画面に遷移しますので，ログインをクリックしてください.
9. パスフレーズの入力画面が出ますので，パスフレーズを入力（下図（8））した上で， OK （下図（9））をクリックしてください．成功しますと WinSCP の画面が表示されファイル転送が可能になります.

3.2.1　WinSCP 用の鍵生成手順

1. WinSCP を起動した後,「ツール」をクリックし,「PuTTYgen を実行」を選択します.
2. PuTTYgen を起動すると,「PuTTY Key Generator」ダイアログボックスが表示されますので,「Load」をクリックします.

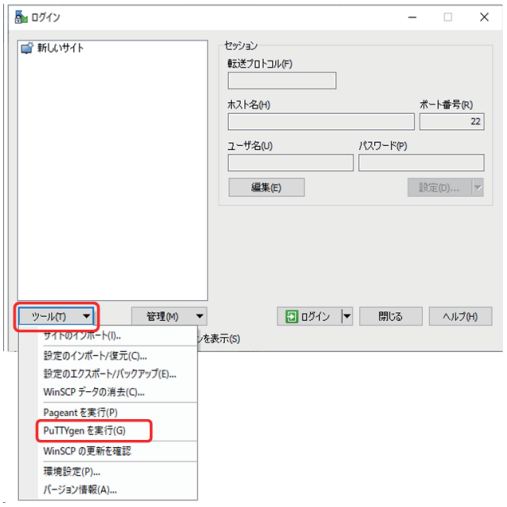


図 5　WinSCP 初期設定画面

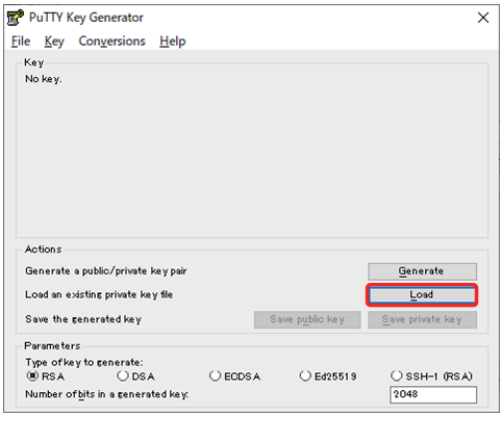


図 6　PuTTY Key Generator 画面

3. ファイルの選択画面が表示されますので，ポータルサイトで作成した秘密鍵「id_rsa_cc」を選択すると，パスフレーズの入力を求められます.
4. パスフレーズの内容が一致すると Notice(情報) メッセージが表示されるので，OK をクリックします.
5.「Save private key」をクリックし，ファイル名を設定します.
6. 設定が完了しましたら，「PuTTY Key Generator」の画面は閉じてください.

4 おわりに

本稿では，ストレージシステムの利用法を紹介しました．ご不明な点，ご質問等ございましたら，お気軽にセンターまでお問い合わせください．問い合わせ先については「利用相談」（https://www.ss.cc.tohoku.ac.jp/consultation/）をご参照下さい.

［大規模科学計算システム］

SX-Aurora TSUBASA でのプログラミング（ベクトル化編）

岡野 進一　工藤 淑裕
日本電気株式会社

概 要

SX-Aurora TSUBASA システムのハードウエア性能を引き出すために重要となるプログラムのベクトル化、並列化に関わるコンパイラの機能、およびプログラミングの際にご留意頂きたい点について、「ベクトル化編」、「並列化編」の 2 回に分けてご紹介します。

SX-Aurora TSUBASA システムでは主なプログラミング言語として、Fortran、C、C++言語が利用できます。本稿では、SX-Aurora TSUBASA システムの Fortran 言語コンパイラである NEC Fortran コンパイラ（以降、単にコンパイラと略す）を用いて、コンパイラが持つ自動ベクトル化機能の特長、ベクトル化プログラミングでの性能向上についてご紹介します。

1. SX-Aurora TSUBASA の Vector Engine

今回、東北大学サイバーサイエンスセンターに導入された SX-Aurora TSUBASA B401-8 システムは、Linux OS 搭載の x86 ホストに、8 個のベクトルコアを内蔵した複数の Vector Engine(以下、VE)を搭載したビックコア、高メモリ帯域、省電力なスーパーコンピュータです。各ベクトルコアは一つの命令で最大 256 個のデータを処理できるベクトル演算器を備えています。

SX-Aurora TSUBASA のベクトル命令には、一般的なスカラプロセッサで導入されている複数のデータを一度に処理できる SIMD 命令に対して、次の特長があります。

- 最大 256 個までの大量のデータを一つのベクトル命令で演算できる
- 一つのベクトル命令で処理するデータの個数を自在に変更できる

SX-Aurora TSUBASA のベクトルコアのハードウエア性能を十分に引き出すためには、個々のベクトルコアの中でベクトル命令を使って効率的に計算するためのベクトル化が大変重要となります。

2. 自動ベクトル化機能

SX-Aurora TSUBASA システムでのベクトル化技法は、SX-ACE システムの場合と基本的には同じですが、今回はスーパーコンピュータを初めて使われる方にもご理解頂けるようベクトル化の基本概念からご紹介します。

2.1. ベクトル化の基本概念

通常の演算命令は、一度に一組のデータを演算できます。このような演算命令をベクトル命令と対比させるためにスカラ命令と呼びます。これに対してベクトル命令は、複数の組のデータに対する演算を一つの命

令で実行できます。

例-1 のプログラムは、配列 B と配列 C を加算し配列 A に代入、配列 E と配列 F を加算し配列 D に代入する DO ループです。

例-1

```
DO I = 1, 100
   A(I) = B(I) + C(I)
   D(I) = E(I) + F(I)
ENDDO
```

例-1 の DO ループ中の加算をスカラ命令、ベクトル命令で実行したときの実行イメージは図-1 のとおりです。一つのスカラ命令は、一度に一組のデータに対する演算処理を行います。これに対して、SX-Aurora TSUBASA のベクトル命令では一つの命令で一度に複数(この場合は 100 個、最大 256 個)のデータに対する演算処理を行うことができます。

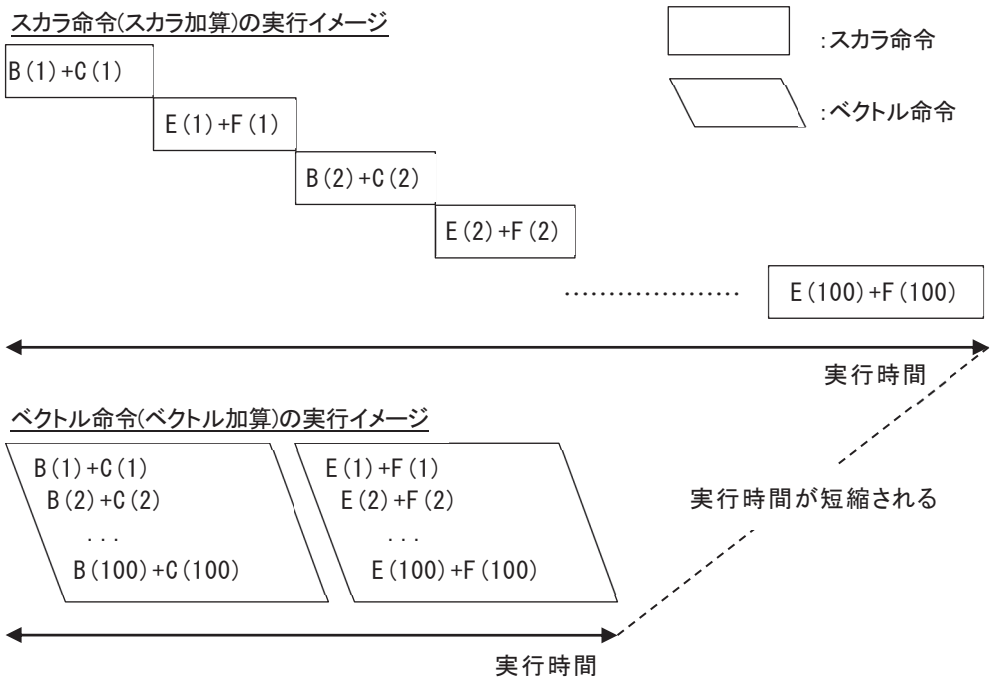


図-1 加算の実行イメージ

ループ中で計算される行列の要素など、規則的に並んだ配列データに対してベクトル命令を適用することをベクトル化(の適用)と呼び、ベクトル化することによって高速な演算が可能となります。

コンパイラの自動ベクトル化機能は、ソースプログラムを解析し、ベクトル命令で実行できる部分を自動的に検出しベクトル化を適用します。

2.2. ベクトル命令の適用例

例-2 は、DO ループが自動ベクトル化されたときのベクトル命令の適用例です。二つの配列のメモリロード、ベクトル加算、メモリストアの四つのベクトル命令で実行されます。

例-2

```
DO I = 1, 100
   C(I) = A(I) + B(I)
ENDDO
```

↓

```
VR1    ←  配列A          (配列Aからベクトルレジスタに100個のデータをロード)
VR2    ←  配列B          (配列Bからベクトルレジスタに100個のデータをロード)
VR3    ←  VR1 + VR2      (100個のデータをベクトル加算)
配列C ←  VR3             (配列Cに100個の演算結果をストア)
                    VRn : ベクトルレジスタ
```

2.3. ベクトル化の対象範囲

自動ベクトル化機能は、表-1 で示すループ、および、ループに含まれる文、データ、演算を対象としてベクトル化を適用します。

表-1　自動ベクトル化の対象

対象	Fortran 言語要素
ループ	配列式、DO ループ、DO WHILE ループ、FORALL ループ
文	代入文、CONTINUE 文、GOTO 文、CYCLE 文、EXIT 文、IF 文、SELECT 構文(CALL 文、入出力文等は不可)
データ型	INTEGER(KIND=4)、INTEGER(KIND=8)、REAL(KIND=4)、REAL(KIND=8)、COMPLEX(KIND=4)、COMPLEX(KIND=8)
演算	加減乗除算、べき算、論理演算、関係演算、型変換、組込み関数(利用者定義演算等は不可)

2.4. データの依存関係

ループにベクトル化を適用するには、「2.3. ベクトル化の対象範囲」で示した対象範囲に加えて、ベクトル化の適用による文、演算の実行順序が変更されても、データの定義・参照順番(データの依存関係)が変わらないことが条件となります。

例-3 は、二つの配列を定義する DO ループです。図-2 は、例-3 の DO ループのベクトル化適用前、すなわち、スカラでの文の実行順序とベクトル化後の文の実行順序のイメージを示しています。

例-3

```
DO I = 1, 100
   A(I) = B(I) + C(I)
   D(I) = E(I) + F(I)
ENDDO
```

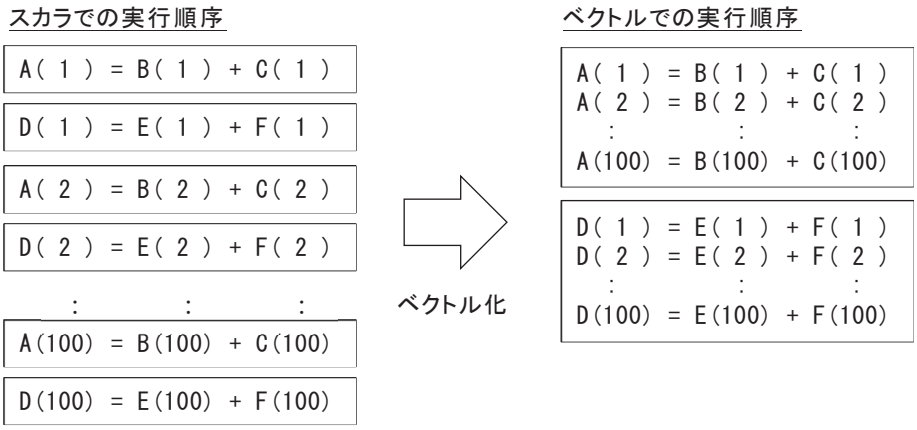


図-2　文の実行順序

スカラでの実行では、A(1)=B(1)+C(1)、D(1)=E(1)+F(1)、A(2)=B(2)+C(2)、...と実行されますが、ベクトル化した場合は一つの命令で複数のデータを処理するため、A(1)=B(1)+C(1)、A(2)=B(2)+C(2)、...、A(100)=B(100)+C(100)、D(1)=E(1)+F(1)、D(2)=E(2)+F(2)、...、D(100)=E(100)+F(100)と配列 B、配列 C の加算、配列 A への代入後、配列 E、配列 F の加算、配列 D への代入が実行されます。ベクトル化した場合、このように文、演算の実行順序が変わります。

このような文の実行順序の変更により、データの依存関係が変わってしまうことがあります。例-4 はベクトル化後にデータの依存関係が変わってしまう例です。以前の繰返しで定義された配列要素や変数を後の繰返しで参照するパターンのとき、図-3 のようにデータの依存関係が変わり正しい結果が得られなくなるため、コンパイラはベクトル化を適用しません。

例-4

```
DO I = 2, N
   A(I+1) = A(I) * B(I) + C(I)
ENDDO
```

図-3 は、例-4 の DO ループの文の実行順序示すイメージです。ベクトル化すると、「2.2. ベクトル命令の適用例」で示した例-2 のように、配列 A のデータをまとめてメモリからロードするため、更新された A の値が使用されず正しい演算結果が得られません。

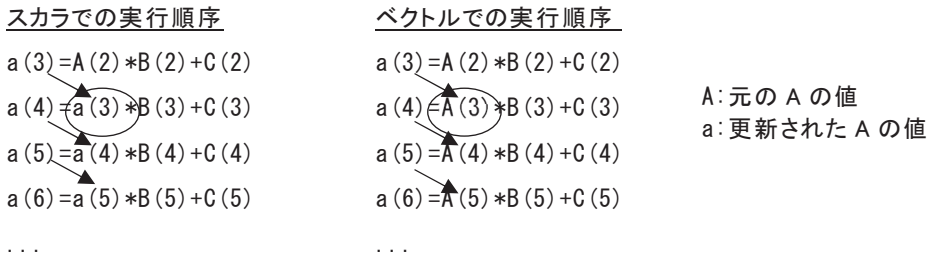


図-3　ベクトル化を阻害するデータの依存関係

例-5 はベクトル化を適用してもデータの依存関係が変わらない例です。

例-5

```
DO I = 2, N
   A(I-1) = A(I) * B(I) + C(I)
ENDDO
```

図-4 は、例-5 の DO ループの文の実行順序を示すイメージです。ベクトル化してもデータの依存関係は変わらないのでコンパイラはベクトル化を適用します。

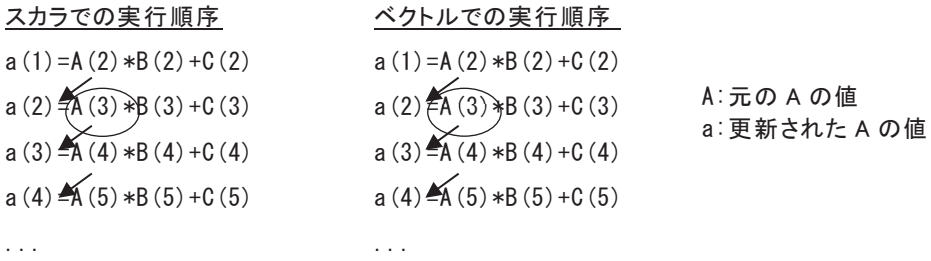


図-4　ベクトル化可能なデータの依存関係

コンパイラの自動ベクトル化機能は、ソースプログラムを解析して、ベクトル命令で実行できる部分を自動的に検出するとともに、必要ならベクトル化に適合するようにプログラムを変形して、ベクトル化を適用できる範囲を広げます。

プログラムのループを記述する際、文をループの繰返しごとに並べたとき、図-3 のような右下向きの矢印の依存関係ができないようにすると、ループにベクトル化を適用でき、ループ内の処理を高速化できる可能性が高まります。

3.　拡張ベクトル化機能

コンパイラの自動ベクトル化機能は、ベクトル化を適用できる範囲を拡げるため、データの依存関係などの理由でベクトル化できないループを変形してベクトル化したり、プログラムを変形することによってベクトル

化の効果をさらに高めたりします。これを拡張ベクトル化機能と呼びます。ここでは、コンパイラの持つ拡張ベクトル化機能のうち主なものをご紹介します。

3.1. 文の入れ換え

「2.4 データの依存関係」の例のようなループは、ループ中の文を入れ換えるとベクトル化できることがあります。コンパイラは、実行結果が正しく保てるのであれば、例-6 に示すように自動的に文を入れ換えてベクトル化を適用します。

例-6

（文の入れ換え前）

```
DO I = 1, 99
   A(I) = 2.0      ! 定義
   B(I) = A(I+1)   ! 参照
ENDDO
```

（文の入れ換え後のイメージ）

```
DO I = 1, 99
   B(I) = A(I+1)      ! 参照
   A(I) = 2.0         ! 定義
ENDDO
```

3.2. ループの一重化

ループの一重化は、多重ループの外側のループと内側のループを一つのループにまとめて、ループの繰返し数を大きくする最適化です。

SX-Aurora TSUBASA のベクトル命令は一つの命令で最大 256 個のデータを処理できます。よって、ベクトル命令で処理するときはできるだけ 256 個ずつ演算した方が命令の実行回数を減らすことができ効率がよくなります。さらに外側ループの繰返し制御のための時間も省くことができます。

自動ベクトル化では最内側ループをベクトル化の対象とします。例-7 では最内側ループの繰返し数は最大でも 100 回と 256 より短いので、このままでは一度に 100 個ずつ処理するようベクトル化されてしまいます。このループに一重化を適用すると、繰返し数を 10,000(=100×100)にでき、256 個ずつ処理できるようになります。

例-7

（ループの一重化前）

```
INTEGER::M, N
PARAMETER(M=100, N=100)
REAL(KIND=8)::A(M,N), B(M,N), C(M,N)
DO I = 1, M
   DO J = 1, N
      A(J,I) = B(J,I) + C(J,I)
   ENDDO
ENDDO
```

（ループの一重化後の変形イメージ）

```
DO IJ = 1, M*N
   A(IJ,1) = B(IJ,1) + C(IJ,1)
ENDDO
```

3.3. ループの入れ換え

多重ループのとき、ループを入れ換えることによりベクトル化できないデータの依存関係の問題が解消されてベクトル化できるようになる場合や、内側のループよりも外側のループの繰返し数が大きく、入れ換えた方が効率がよいと判断された場合には、例-8 のようにコンパイラが自動的にループを入れ換えてベクトル化を適用します。

例-8

（ループの入れ換え前）

```
DO J=1, 1000
   DO I = 1, 999
      A(I+1,J) = A(I,J) + B(I,J)
   ENDDO
ENDDO
```

（ループの入れ換え後のイメージ）

```
DO I = 1, 999
   DO J = 1, 1000
      A(I+1,J) = A(I,J) + B(I,J)
   ENDDO
ENDDO
```

なお、入れ換えた際の効率は、ハードウエア特性、およびループ内での処理内容や配列要素のアクセス方法(連続アクセス、飛びアクセス)など、さまざまな要因により変わります。このため、SX-ACE ではループ入れ換えされていたループが、SX-Aurora TSUBASA ではループ入れ換えの対象とならない場合があります。

3.4. 条件ベクトル化

条件ベクトル化とは、一つのループに対してあらかじめ二つ以上の命令コードを用意しておき、実行時に最も効率よく実行できる命令コードを選択して実行するベクトル化です。

例-9 は、データの依存関係がベクトル化に適合しているかどうかコンパイル時に不明であったとき、ベクトル化した命令コードとベクトル化しない命令コードの両方を用意しておき、プログラムを実行するときそのどちらかを選択して実行する条件ベクトル化の例です。

例-9

（条件ベクトル化前）

```
DO I = N, N+10
   A(I) = A(I+K) + B(I)
ENDDO
```

（条件ベクトル化後のイメージ）

```
IF (K.GE.0.OR.K.LT.-10) THEN
!NEC$ IVDEP              ! ベクトルでの実行
   DO I = N, N+10
      A(I) = A(I+K) + B(I)
   ENDDO
ELSE
   DO I = N, N+10        ! スカラでの実行
      A(I) = A(I+K) + B(I)
   ENDDO
ENDIF
```

例-9 の条件ベクトル化後のイメージの IF 文が実行時にコードを選択するための判定文で、THEN ブロックはベクトルで実行するコード、ELSE ブロックはスカラで実行するコードです。IF 文の条件式に現れる

「K.GE.0」が真のときは例-5 と同じようにデータの依存関係が変わりません。また、「K.LT.-10」のときには、「DO I=N,N+10」であることから常に I≠I+K が成り立ちデータの依存関係がありません。よって、条件式が真のときベクトルでの実行(THEN ブロック)となります。逆に、IF 文の条件式が偽であるとき、「2.4 データの依存関係」の例-4、図-3 で示したようなベクトル化後にデータの依存関係が変わってしまう(右下向きの矢印ができてしまう)のでスカラでの実行(ELSE ブロック)となります。

例-9 はデータの依存関係に着目した条件ベクトル化(依存関係による条件ベクトル化)です。他の条件ベクトル化として、ループの繰返し数による条件ベクトル化も行います。

3.5. マクロ演算の認識

次のようなパターンは、変数や配列要素が繰り返しにまたがって定義・引用されるため、本来はベクトル化できませんが、コンパイラが特別なパターンであることを認識し，専用のベクトル命令を用いることで，ベクトル化を行います。

例-10 総和

```
SUM = 0.0
DO I=1, N
   SUM = SUM + A(I)
END DO
```

例-11 最大値・最小値

```
DO I=1, N
   IF (XMAX .LT. X(I)) THEN
     XMAX = X(I)
   END IF
END DO
```

例-12 圧縮・伸長

```
J = 0
DO I=1, N
   IF (X(I) .GT. 0.0) THEN
      J = J + 1
      Y(J) = Z(I)
   END IF
 END DO
```

例-13 漸化式

```
DO I=1, N
    A(I) = A(I-1) * B(I) + C(I)
END DO
```

4. 基本的な使い方

この章では、コンパイラの一般的な使い方について説明します。

4.1. コンパイラの起動

SX-Aurora TSUBASA システム用の Fortran コンパイラのコマンドは nfort です。一般的なコンパイラ同様、例-14 のように使用します。

例-14

$ nfort [コマンドラインオプション…] 入力ファイル…

4.2. コマンドラインオプション

代表的なコマンドラインオプションは、GNU コンパイラなど一般的なコンパイラと同じなので、Makefile などを大きく修正することなく使用できます。一方、SX-ACE システム用のコンパイラのコマンドラインオプションからは多くが変更されているため、ユーザーズガイドの付録 B の対応表を参考にして、適宜修正してください。

以下では、最適化やチューニングにおける主なコマンドラインオプションについて紹介します。

-O*n*

自動ベクトル化、最適化のレベルを指定します。各レベルの説明を表-2 に示します。

表-2　自動ベクトル化のレベル

-O4	最大レベルの自動ベクトル化を適用
-O3	高度なレベルの自動ベクトル化を適用
-O2	既定レベルの自動ベクトル化を適用（既定値）
-O1	副作用のない自動ベクトル化を適用
-O0	ベクトル化、最適化を適用しない

-mparallel

自動並列化機能を適用します。詳細は、「並列化編」でご紹介します。

-fopenmp

OpenMP 機能を適用します。詳細は、「並列化編」でご紹介します。

-finline-functions

自動インライン展開機能を適用します。

-report-all

オプションリスト、診断メッセージリスト、編集リストを出力します。

オプションリストには、コンパイラオプションのコンパイル時の状態(有効、無効、値など)が出力されます。

例-15　　オプションリストの例

```
NEC Fortran Compiler (3.0.7) for Vector Engine Mon Aug 17 13:25:29 2020
FILE NAME: fft.f90

  COMPILER OPTIONS : -report-option

OPTIONS DIRECTIVE: -O4

PARAMETER :

Optimization Options :
  ....
  -On                          : 4
  -fargument-alias             : disable
  -fargument-noalias           : enable
  -fassociative-math           : enable
  ....
```

診断メッセージリストや編集リストはプログラムに対してコンパイラがどのような最適化を適用したか、および、適用できなかった理由等が出力されます。診断メッセージリストは、コンパイル時に標準エラー出力に出力された診断メッセージがリストとなって出力されます。編集リストは以下のような形式で出力されます。

例-16　　編集リストの例

```
NEC Fortran Compiler (3.0.7) for Vector Engine Mon Aug 17 15:00:01 2020
FILE NAME: a.f90

PROCEDURE NAME: SUB
FORMAT LIST

 LINE   LOOP   STATEMENT
```

```
1:            SUBROUTINE SUB(A, B, N, M)
2:            INTEGER::N, M
3:            REAL(KIND=8)::A(M, N), B(M, N)
4:    +------>  DO J=1,M
5:    |V----->    DO I=1, N
6:    ||           A(I,J) = A(I,J) + B(I,J)
7:    |V-----     ENDDO
8:    +------   ENDDO
9:            END SUBROUTINE
```

4.3. コンパイラ指示行

コンパイラは指定されたコンパイルオプションに応じて、最適なベクトル化、および最適化を適用しますが、コンパイル時にソースプログラムからコンパイラが自動で認識できない情報(変数の値やループの繰り返し数など)のため、本来適用すべきベクトル化や最適化ができない場合があります。そのような場合、コンパイラ指示行を指定することにより、コンパイル時に必要な情報を補い、ベクトル化や最適化を適用できるようになります。

IVDEP 指示行

コンパイル時にコンパイラが自動的に依存関係を解析しループの自動ベクトル化を試みますが、コンパイル時に依存関係が不明な場合、ループを自動ベクトル化しません。このとき、プログラマがプログラムに依存関係がないことが分かっている場合に IVDEP 指示行を指定すると、コンパイラはベクトル化が不可となる依存関係はないと仮定してベクトル化を試みます。

例-17

```
SUBROUTINE SUB(A,B,C,N,K)
   REAL::A(N), B(N), C(N)
   INTEGER::N,K,I

!NEC$ IVDEP
   DO I=1,N
      A(I+K) = A(I) + B(I)
   END DO

END SUBROUTINE SUB
```

NOVECTOR 指示行

自動ベクトル化の適用から除外するためには、NOVECTOR 指示行を使用します。依存関係はなく、自動ベクトル化は行われたが、実際のループの繰り返し数が非常に少なく、ベクトル化しない場合の方が高速な場合やベクトル化による演算結果の誤差が気になる場合などに使用します。

例-18

```
SUBROUTINE SUB(A,B,C,N,K)
   REAL:: A(N), B(N), C(N)
   INTEGER::N,K,I

!NEC$ NOVECTOR
   DO I = 1, M
      A(I+K) = A(I) * B(I)
   ENDDO

END SUBROUTINE SUB
```

OUTERLOOP_UNROLL 指示行

多重ループの場合、外側のループに対して、OUTERLOOP_UNROLL 指示行を指定することで、外側のループを内側のループ内部に展開(アンロール)することにより、内側ループのインデックスのみを使用するロードやストアの命令の実行数を減らすことができ、高速化できます。

例-19

```
!NEC$ OUTERLOOP_UNROLL(4)
   DO J = 1, M
      DO I = 1, N
         A(I,J) = B(I,J) + C(I)     ! C(I)のロードが1/4に減る
      ENDDO
   ENDDO
```

nfdirconv コマンド

コマンドラインオプションと同様、多くのコンパイラ指示行も、SX-ACE 用のコンパイラである FORTRAN90/SX や Fortran 2003 コンパイラのコンパイラ指示行から変更されています。

nfdirconv コマンドは、SX-ACE システム用のコンパイラである FORTRAN90/SX や Fortran 2003 コンパイラ用に記述されたコンパイラ指示行を、SX-Aurora TSUBASA システム用の Fortran コンパイラのコンパイラ指示行に変換するツールです。

使い方は、例-20 のとおりで、入力ファイルの代わりにディレクトリを指定した場合、ディレクトリ配下のすべてのファイルのコンパイラ指示行をまとめて変換できます。

例-20

```
$ nfdirconv [オプション …] 入力ファイル …
```

4.4. ベクトル化による演算結果への影響

この節では、SX-Aurora TSUBASA においてプログラミングを行う際に留意すべき点として、ベクトル化の最適化を行った場合とベクトル化を行わなかった場合で、演算結果が誤差範囲で異なる場合があります。

- 最適化・ベクトル化による演算順序の変更や除算の乗算化により、情報落ちや桁落ち、丸め誤差などが変わるため
- ベクトル化された数学関数では、高速にベクトル計算できるようスカラ版の数学関数と異なる計算アルゴリズムを使用しているため
- ベクトル積和演算(fused multiply-add, FMA)が使用された場合、途中の積算結果を丸めずに加算が行われるため、使用しない場合に対して異なる演算結果となる可能性があるため

このような誤差が気になる場合には、NOVECTOR 指示行を指定してループがベクトル化されないようにしたり、NOFMA 指示行を指定してベクトル融合積和演算が行われないようにしてください。

4.5. 実行時環境変数

この節では、プログラムの実行時に便利ないくつかの実行時環境変数を説明します。

VE_FORT_UFMTENDIAN

SX-Aurora TSUBASA は SX-ACE と異なりリトルエンディアンのシステムです。SX-ACE などのビッグエンディアンのシステムの外部ファイルを書式なし入出力するとき、エンディアン変換が必要となります。そのようなとき、この環境変数で、ビッグエンディアンの外部ファイルが接続されている装置番号を指定します。

例-21　すべての外部ファイル装置に対してビッグエンディアンとする指定

```
$ export VE_FORT_UFMTENDIAN=ALL
```

例-22　外部ファイル装置 10、11に対してビッグエンディアンとする指定

```
$ export VE_FORT_UFMTENDIAN=10,11
```

例-23　外部ファイル装置 10、11、12 に対してビッグエンディアンとする指定

```
$ export VE_FORT_UFMTENDIAN=10-12
```

例-24　外部ファイル装置 10、11、12 以外をビッグエンディアンとする指定

```
$ export VE_FORT_UFMTENDIAN=big;little:10-12
```

VE_FORT_EXPRCW

2GB(ギガバイト)を超える記録(レコード)を扱うために拡張されたフォーマットを使用する書式なし入

出力の外部ファイル装置を指定します。

例-25　　外部ファイル装置 10 に対して指定

```
$ export VE_FORT_EXPRCV=10
```

5. 性能解析

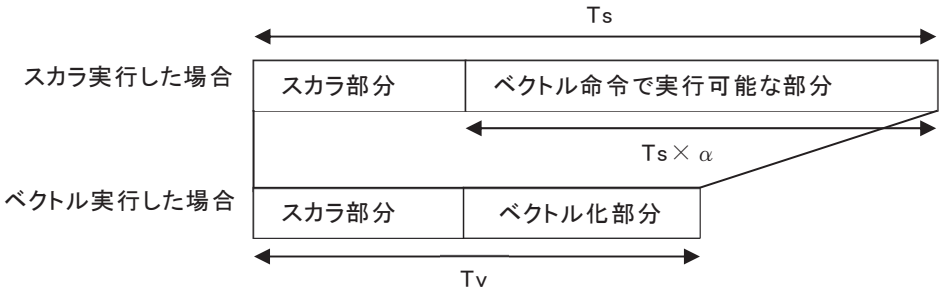
SX-Aurora TSUBASA においては、プログラムに含まれるループをできるだけベクトル命令を使って、データを最大ベクトル長である 256 個ずつ実行できているかどうかが高速化のポイントです。

本セクションでは、プログラムの性能を分析するための性能値、その取得方法、ベクトル化が十分できていない手続を絞り込む方法などを説明します。

5.1. ベクトル化率

プログラムをスカラ命令だけで実行させた場合の実行時間に占めるベクトル命令で実行可能な部分の時間の割合をベクトル化率と呼びます。一般にベクトル化率を正確に求めることは困難であるため、ベクトル化率の近似値としてベクトル演算率を用います。

ベクトル演算率は、プログラムで処理された全演算数に占めるベクトル命令で処理された演算数の割合を求めたものです。SX-Aurora TSUBASA では、このベクトル演算率を大きくすることを目標にチューニングしてください。ベクトル演算率は、プログラム中のループをベクトル化することにより実行性能が向上しますが、ベクトル演算率が 50%程度では、スカラ実行時の高々2 倍の性能にしかならないことは、図-5 からもわかると思います。一般にはベクトル演算率が 98%以上を目指すことになります。ベクトル演算率は、以降で示すプログラム実行解析情報、簡易性能解析機能/Ftrace Viewer で参照できます。



Ts：スカラ実行したときの実行時間　　α：ベクトル化率

Tv：ベクトル実行したときの実行時間

図-5 ベクトル化率

5.2. プログラム実行解析情報

プログラム実行解析情報は、プログラムの実行時に参照される環境変数 VE_PROGINF に YES、または、DETAIL が設定されているとき、プログラムの実行終了時に標準エラー出力ファイルに出力されます。この情報から、プログラムがハードウエアの性能を十分に引き出しているか否かを判断できます。

例-26 にプログラム実行解析情報の出力例を示します。

例-26

```
            ******  Program Information  ******
Real Time (sec)                 :          204.076110   経過時間
User Time (sec)                 :          203.706817   ユーザ時間
Vector Time (sec)               :          197.623752   ベクトル命令実行時間
Inst. Count                     :        38596814372    全命令実行数
V. Inst. Count                  :        13465836887    ベクトル命令実行数
V. Element Count                :      2957231889428    ベクトル命令実行要素数
V. Load Element Count           :       997524789907    ベクトル命令ロード要素数
FLOP Count                      :      1776569208614    浮動小数点データ実行要素数
MOPS                            :          18087.515129 MOPS値(ユーザ時間)
MOPS (Real)                     :          18053.533350 MOPS値(経過時間)
MFLOPS                          :           8721.924006 MFLOPS値(ユーザ時間)
MFLOPS (Real)                   :           8705.537759 MFLOPS値(経過時間)
A. V. Length                    :            219.609959 平均ベクトル長
V. Op. Ratio (%)                :             99.317880 ベクトル演算率
L1 Cache Miss (sec)             :              5.637238 L1キャッシュミス時間
CPU Port Conf. (sec)            :              0.125939 CPUポート競合時間
V. Arith Exec. (sec)            :             29.765092 ベクトル演算実行時間
V. Load Exec. (sec)             :            163.530245 ベクトルロード実行時間
VLD LLC Hit Element Ratio (%)   :             58.115252 ベクトル要素LLCヒット率
Power Throttling (sec)          :              0.000000 電力要因HW停止時間
Thermal Throttling (sec)        :              0.000000 温度要因HW停止時間
Memory Size Used (MB)           :           5376.000000 最大メモリ使用量
```

プログラム実行解析情報のベクトル演算率が 98%未満のとき、プログラムを調べ、ベクトル化できていないループがないかなどを調査します。

5.3. 簡易性能解析機能／Ftrace Viewer

プログラムの規模が大きいとき、プログラム内のコードすべてについてループのベクトル化状況を調べるのは作業効率がよくありません。ベクトル性能を引き出すには、実行時間が長く、ベクトル演算率が低い手続に絞って調べることが効果的です。この絞り込みには簡易性能解析機能、または Ftrace Viewer が有効です。

Ftrace Viewer では、手続ごとの実行回数、実行時間、MFLOPS 値、ベクトル演算率、LLC ヒット要素率などの様々な性能値に加えて、それらをグラフ(Function Metric Chart)表示し、視覚的にプログラムの性能を分析できます。

図-6 は、実行時間とベクトル演算率を重ねて表示したグラフです。これを参照し、実行時間が長く、ベクトル演算率の低いものを探し、その中のループのベクトル化状況を調べます。

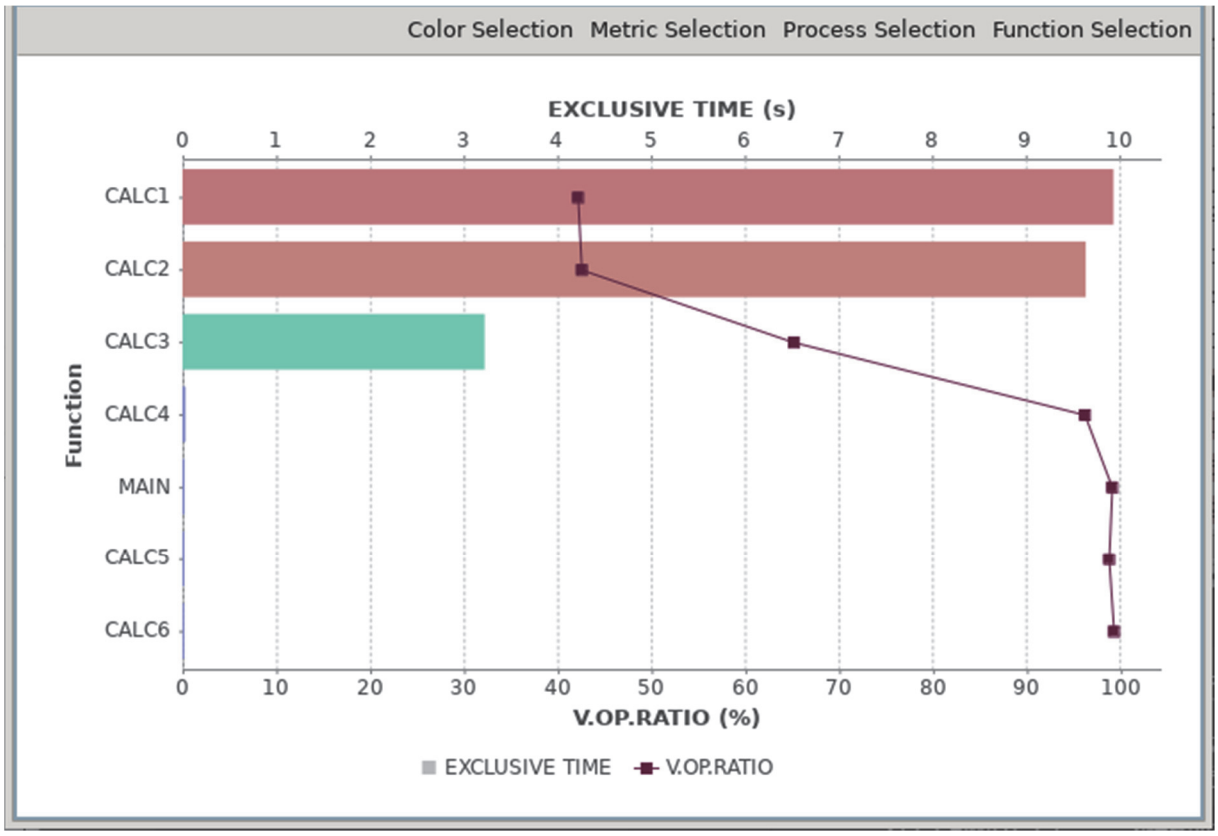
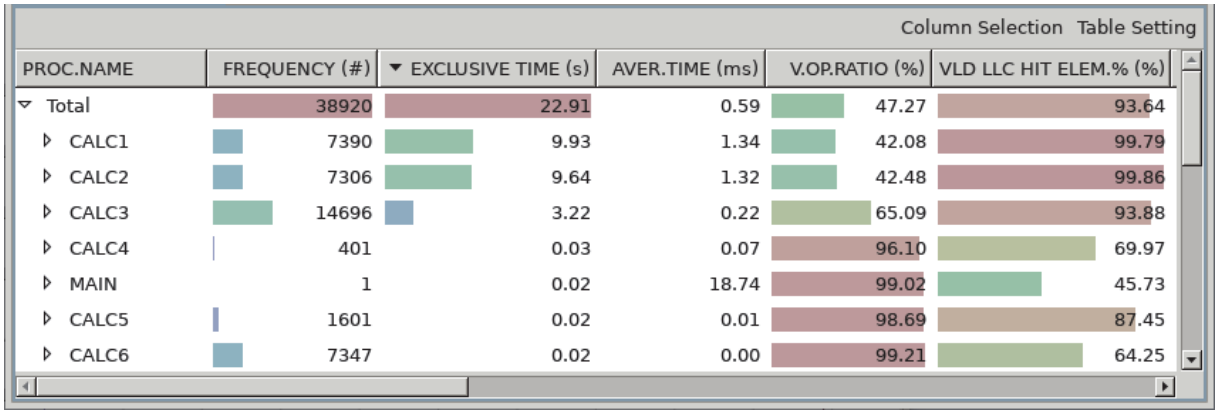


図-6 Function Metric Chart

図-6 では、縦軸が手続名、上横軸が実行時間(EXCLUSIVE TIME、単位は秒)、下横軸がベクトル演算率(V.OP.RATIO、単位は%)です。実行時間は棒グラフ、ベクトル演算率は折れ線グラフで表示されています。

このグラフから、実行時間が長くベクトル演算率が低い手続は「calc1」と「calc2」であることが分かります。「calc1」も「calc2」も実行時間が長く、かつ、ベクトル演算率も低いため、ベクトル演算率を高めるという観点からのチューニングにより高速化が期待できます。また、「calc3」についてもベクトル演算率が十分に高いとは言えないため、チューニングの余地はありそうです。その他の手続については、ベクトル演算率も高く、実行時間も非常に短くなっているため、これ以上のチューニングの必要はなさそうです。

個々のサブルーチン、手続の詳細な性能値を調べたいとき、グラフと一緒に表示されるテーブル(Function Table)を参照します。



PROC.NAME	FREQUENCY (#)	EXCLUSIVE TIME (s)	AVER.TIME (ms)	V.OP.RATIO (%)	VLD LLC HIT ELEM.% (%)
Total	38920	22.91	0.59	47.27	93.64
CALC1	7390	9.93	1.34	42.08	99.79
CALC2	7306	9.64	1.32	42.48	99.86
CALC3	14696	3.22	0.22	65.09	93.88
CALC4	401	0.03	0.07	96.10	69.97
MAIN	1	0.02	18.74	99.02	45.73
CALC5	1601	0.02	0.01	98.69	87.45
CALC6	7347	0.02	0.00	99.21	64.25

図-7　Function Table

図-7 から、手続「calc1」の実行回数(FREQUENCY)は 7390 回、一回当たりの実行時間(AVER.TIME)は 1.34 ミリ秒で総実行時間は 9.93 秒だったことがわかります。また、手続「calc2」は実行回数(FREQUENCY)は 7306 回、一回当たりの実行時間(AVER.TIME)は 1.34 ミリ秒で総実行時間は 9.64 秒だったことがわかります。さらに、ベクトル演算率(V.OP.RATIO)は手続「calc1」が 42.08%、手続「calc2」が 42.48%だったこともわかります。

Ftrace Viewer で性能値を参照するには、コンパイラオプション-ftrace を指定してコンパイルされたプログラムを実行して、解析情報ファイル(ファイル名 ftrace.out.*.*)を出力することが必要です。

6.　おわりに

以上、NEC Fortran コンパイラの自動ベクトル化機能を中心にご紹介させて頂きました。SX-Aurora TSUBASA のコンパイラは、他にも本稿でご紹介できなかった種々のベクトル化機能を持っています。詳細につきましては、マニュアル「SX-Aurora TSUBASA FORTRAN コンパイラ ユーザーズガイド」をご参照ください。

皆様が SX-Aurora TSUBASA をご利用になる上で、本稿が多少なりともお役に立てれば幸いです。

参考文献

[1] NEC Aurora Forum　https://www.hpc.nec/

[2] SX-Aurora TSUBASA Fortran コンパイラ ユーザーズガイド 日本電気株式会社 G2AF02
https://www.hpc.nec/documentation

[3] SX-Aurora TSUBASA Ftrace Viewer ユーザーズガイド 日本電気株式会社 G2AT01
https://www.hpc.nec/documentation

［共同研究成果］

複数領域 FDTD（時間領域差分）法の高速化

有馬 卓司，宇野 亨
東京農工大学　大学院工学研究院　先端電気電子部門

本報告では，電磁界解析手法の一つである FDTD 法の高速計算手法について述べる．FDTD 法は差分法を用いており計算コストが比較的多いのが特徴である．また，FDTD 法の解析空間を複数領域に分け，領域同士を数学的に結合させる複数領域 FDTD 法も提案されているが，この手法も結合の計算に多くの計算が必要であり，有効性が限られていた．本報告では，この複数領域 FDTD 法に注目しその高速化手法を示す．高速化には，画像処理などで用いられている四分木構造の考え方を活用している．

1.　複数領域 FDTD 法

電波は，通信だけでなく電子レンジや自動車の衝突防止レーダなど様々なところで使用されている．これら電波を用いた機器の開発においては，計算機を用いた数値解析を活用する事で実験や試作を減らすことが出来るので有用である．電磁界解析の主要な方法の一つに FDTD（Finite Difference Time Domain ： 時間領域差分）法がある[1]．この手法は，電磁界を支配するマクスウエルの方程式を直接差分し，電界および磁界を求める手法である．差分法は一般的に計算量が多くなることが知られており，FDTD 法も多くの計算機資源が必要となる．FDTD 法の計算量を削減する手法がいくつか提案されている．計算削減手法の一つに複数領域 FDTD 法[2]が提案されている．この手法の概念を図 1 に示す．

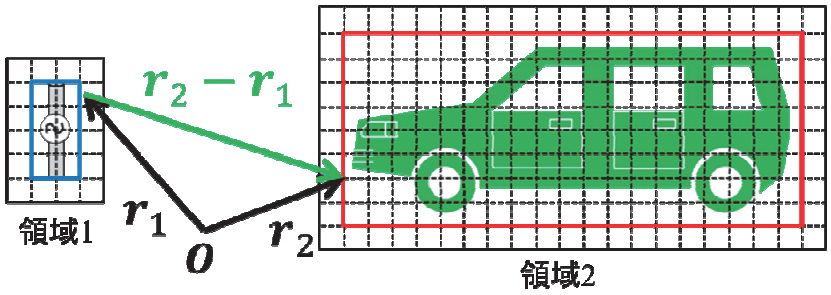


図 1　複数領域 FDTD 法の概念

図 1 に示すように複数領域 FDTD 法においては，解析空間中で波源や散乱体のない自由空間の計算を FDTD 法では行わず，数学的にその結合を評価する手法である．この手法は，FDTD 法自体の計算領域を大幅に減らせる事から計算量削減が期待できる．一方，複数領域同士の結合には複雑な積分が必要になる．ここでは例として，複数領域 FDTD 法において領域 2 における電磁界を，領域 1 の電磁界から数学的手法で求める方法を説明する．まず，図 1 の領域 1 内に仮想的に設置された青枠で示される閉曲面上の電界および磁界より，等価電流 $\mathbf{J}_1$ および等価磁流 $\mathbf{M}_1$ を求める．そして求まった，等価電磁流と自由空間のグリーン関数 G を用いて，領域 2 における電界は(1)式のように求められる．

$$\mathbf{E}_2(\mathbf{r}_2,\omega)=\int_{S_1}\left[-j\omega\mu\mathbf{J}_1(\mathbf{r}_1,\omega)G(\mathbf{R})-\mathbf{M}_1(\mathbf{r}_1,\omega)\times\nabla_1 G(\mathbf{R})+\left\{\hat{\mathbf{n}}_1\cdot\mathbf{E}_1(\mathbf{r}_1,\omega)\right\}\nabla_1 G(\mathbf{R})\right]\mathrm{d}S_1 \qquad (1)$$

(1)式中で，$\mathbf{R}$ は領域 1 から領域 2 に向かうベクトルである．また，$\hat{\mathbf{n}}_1$ は領域 1 に設定した仮想閉曲面の法線ベクトルである．この面積分を行えば，領域 1 で生じた電磁界を基に領域 2 において

正しい電界を再構成できることを示している．この面積分によって電界を再構成する原理は等価定理と呼ばれている．磁界についても同様に再構成できる．一方，(1)式の計算は，領域 2 のある点の電界を求めるために，領域 1 に設置した仮想閉曲面全体にわたり積分を実施する必要がある．すなわち，図 1 の領域 2 に示す赤枠の仮想的な閉曲面上のすべての電磁界を再構成するには膨大な計算量となることが分かる．

複数領域 FDTD 法による計算例を示す．計算モデルは図 2 に示すように，一つの領域に全長 150mm のダイポールアンテナがあり，その領域の中心から 84 mm 離れたところに完全導体があり，この領域をアンテナとは別の領域としている．(1)式の等価定理を計算するための仮想閉曲面の大きさ，および電界を再構成するための仮想閉曲面の大きさはそれぞれの領域において，36mm×36mm×210mm および 108mm×108mm×108mm としている．FDTD 法のメッシュサイズは各方向にそれぞれ 3mm とした．図 3 に解析結果を示す．この結果は図 2 中に示されている”Observed point”における電界を通常の FDTD 法と複数領域 FDTD 法で計算した結果である．図 3 において，赤の点線で示されているものは通常の FDTD 法を用いて解析領域を分けずに計算した結果である．緑の実線で示されている結果は複数領域 FDTD 法での計算結果である．通常の FDTD 法と同様の結果が得られているが，計算時間を進めていくと不安定になる事が分かる．青の実線で示された結果は，安定化のフィルターを通したものである．このように，安定化のフィルターが必要ではあるが，通常の FDTD 法と同様の結果がえられる．一方計算時間であるが，このモデルでは領域と領域の距離が近いために複数領域 FDTD 法のメリットが現れず複数領域 FDTD 法の方が多くの計算時間を必要とする．

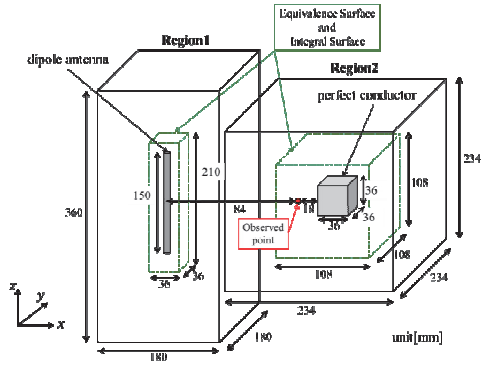


図 2　解析モデル

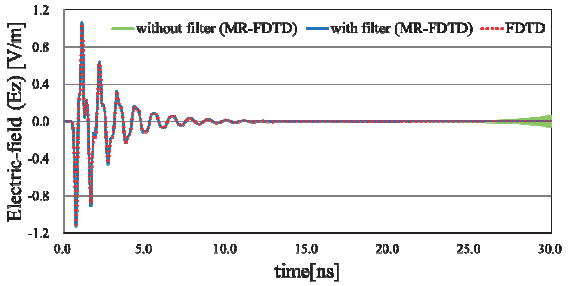


図 3　複数領域 FDTD 法の解析結果

2.　四分木構造を用いた複数領域 FDTD 法の計算量削減について

2.1 複数領域 FDTD 法の計算手順

本節では複数領域 FDTD 法の高速計算化手法を示す．まず通常の複数領域 FDTD 法における(1)式の具体的な計算方法を示す．(1)式は面積分であるので，この面積分を評価するために仮想閉曲面上の FDTD 法で用いているメッシュの数だけ和を求めることになる．FDTD 法のメッシュおよび仮想閉曲面のイメージおよび仮想閉曲面上の電界の配置を図 4 に示す．例えば，仮想閉曲面の大きさをそれぞれの領域で 50 メッシュ×50 メッシュ×6 面とすると要素の数は 15,000 程度となる．これを別の領域に同様の大きさの閉曲面上で計算するとなると，この 2 乗になり大きな計算量となる．また，この仮想閉曲面上で電磁界を再構成するには，電界および磁界についてそれぞれ面に沿う 2 方向の成分が必要となる．

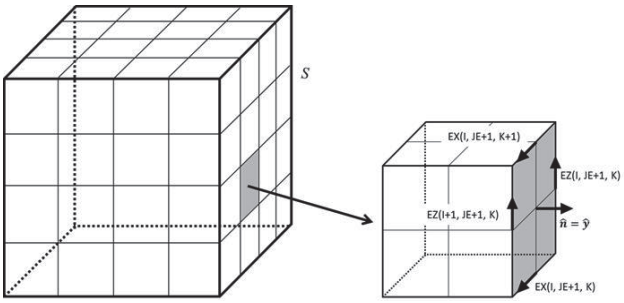


図 4　仮想閉曲面 s と面上の電界および法線ベクトル

2.2 四分木構造を用いた複数領域 FDTD 法の計算量削減手法

本節では，四分木構造[3][4]を用いた複数領域 FDTD 法の計算量削減手法について述べる．前節で述べたように，複数領域 FDTD 法の計算においては等価定理の面積分を効率よく計算することが重要となる．単純に計算量を減らすことを考えれば，面積分に用いるメッシュのサイズを大きくすることで，面積分自体の計算量を減らすことが出来る．しかし，単純にメッシュサイズを大きくすると電磁界を再構成した際の精度の低下を招くことは容易に想像できる．一方，仮想閉曲面上の電磁界は変化の激しい領域と変化が緩やかな領域がある．この特性に注目した計算量削減手法について述べる．まず，この手法では注目するメッシュと周りのメッシュの電磁界の変化を確認し変化が緩やかであると確認された際は，四分木構造を利用して(1)式の面積分を計算するためのメッシュを仮想的に拡大し面積分を実行する．四分木構造とは注目するセルの周りのセルを正方形になるように拡大していく手法であり画像圧縮などで広く用いられている．画像圧縮では四分木面構造を用いる事で画質をある程度保ったまま圧縮できることが知られている．積分を計算するためのメッシュを仮想的に拡大しているので，拡大したメッシュ上で電磁界の値は各方向成分それぞれ一つとなる．このイメージを図 5，6 に示す．

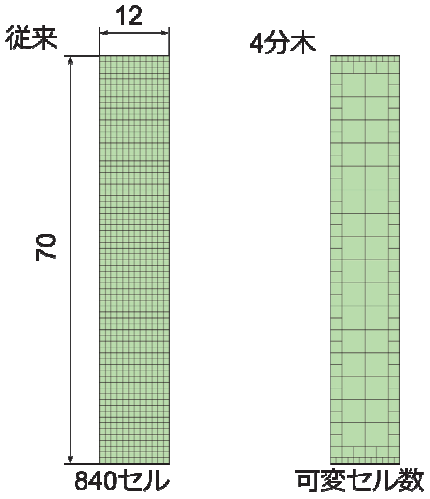


図 5　従来手法と提案手法のメッシュ

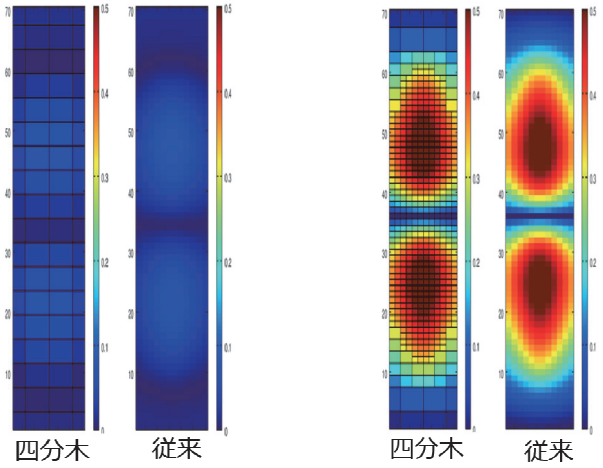


図 6　変化が緩い時のメッシュと急な時のメッシュ

図 5 は，(1)式の面積分を計算するための従来のメッシュと提案手法である四分木構造により仮想的にそれぞれのメッシュを拡大した例を示している．このように，一律にメッシュを拡大するのではなく，ある領域は細かくある領域は粗くなっていることが分かる．また，本提案手法においては，メッシュの拡大基準について，四分木構造内の標準偏差をタイムステップごとに計算し，その標準偏差が事前に設けた基準より小さいければメッシュを拡大し，基準よりも大きい時はそれ以上メッシュを拡大しないとしている．図 6 に実際にメッシュを拡大した例を示している．こ

のように電磁界の変化が緩くなると仮想的にメッシュを拡大し（図 6 左），逆に変化が急激な時はメッシュサイズは比較的細かくなる（図 6 右）．

3．　四分木構造を用いた複数領域 FDTD 法による解析結果

本節では提案手法の有効性を確認するために図 2 に示したモデルと同様のモデルの解析を行った．領域同士の間隔は 300mm とし一つの領域にダイポールアンテナがあり，離れた領域で電界を観測する．(1)式の等価定理を計算するための仮想閉曲面の大きさ，および電界を再構成するための仮想閉曲面の大きさはそれぞれの領域において，120mm×120mm×300mm および 450mm×450mm×450mm としている．FDTD 法のメッシュサイズはダイポールアンテナがある領域では，各方向にそれぞれ 3mm とし，電界を観測する領域においては各方向にそれぞれ 9mm とした．なお，本計算において電界を観測する領域に散乱体は配置せず真空とした．図 7 に解析結果を示す．

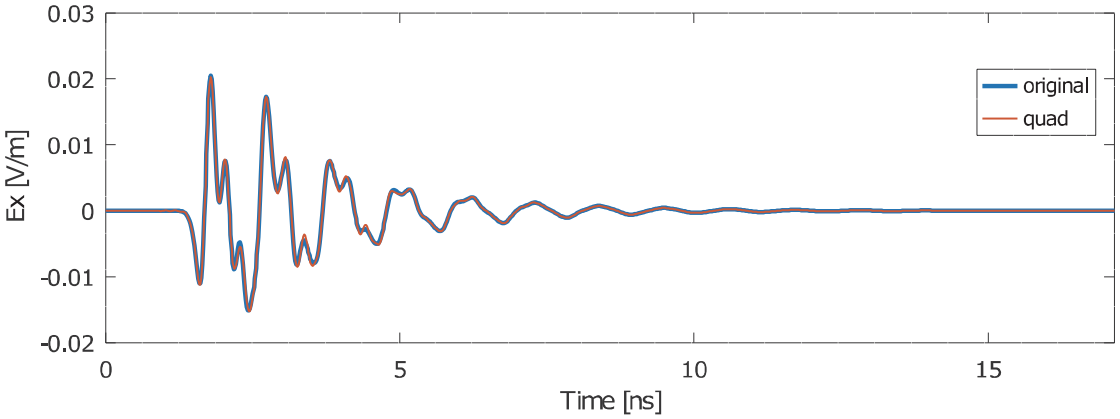


図 7　従来手法と通常の FDTD 法の結果

図 7 に示すように提案手法と通常の FDTD 法を用いた結果はよく一致しており精度が高いことが分かる．計算時間であるが，通常の複数領域 FDTD 法は東北大学設置の SX-ACE 1 ノードを用いて約 28 分必要なのに対し，提案手法を用いた際は約半分の約 15 分で計算が終わっている．以上より提案手法を用いると精度をほぼそのままに計算時間を半減できることが分かった．

4．　まとめ

本研究では，複数領域 FDTD 法においてその計算時間が多く必要である問題に対し計算量削減手法を示した．この手法は，電磁界の急激に変化する領域と比較的変化が緩やかな領域に分けて，面積分を計算するための仮想的なメッシュの大きさを拡大する方法であった．このメッシュの拡大には，四分木構造の考え方を利用し，電磁界の変化が緩やかな領域にメッシュを大きく改題した．計算結果より，提案手法を用いる事で精度をほぼそのままに計算時間を半減できることが分かった．本手法は，FDTD 法の様々な問題に活用できると考える．

謝辞

本研究は，東北大学サイバーサイエンスセンターのスーパーコンピュータを利用することで実現することができた．また，研究にあたっては同センター関係各位に有益なご指導とご協力をいただいた．ここに謝意を示す．

参考文献

[1] 宇野亨，何一偉，有馬卓司，数値電磁界解析のための FDTD 法，コロナ社，2016
[2] J. M. Johnson, Y. Rahmat-Samii, MR/FDTD: A Multiple-Region Finite-Difference

Time-Domain Method, Microwave and Optical Technology Letters, vol. 14, no.2, pp.101-105, February 1997
[3] R. A. Finkel and J. L. Bentley, Quad trees a data structure for retrieval on composite keys,″ Acta Informatica, vol. 4, pp. 1-9, 1974/03/01 1974
[4] G. J. Sullivan and R. L. Baker, Efficient quadtree coding of images and video, IEEE Transactions on Image Processing, vol. 3, no. 3, pp. 327-331, May 1994

［共同研究成果］

直接数値解析を用いた仮想拡張した環状流路における亜臨界遷移の研究

— 局在乱流の間欠構造形成と DP 普遍性の探索 —

塚原 隆裕†・ 竹田 一貴‡

† 東京理科大学　理工学部　機械工学科
‡ 東京理科大学大学院　理工学研究科　機械工学専攻

円管内や平行平板間などの壁乱流に見る亜臨界遷移，そのメカニズムと普遍則の解明には理論的アプローチが困難で，近年は大型ベクトル並列計算機を活用しての直接数値シミュレーション（DNS）によるアプローチが可能となってきた．本研究では，同軸二重円筒内の環状クエット流を対象にして，乱流維持限界の下臨界レイノルズ数付近で観察される局在乱流の間欠構造に注目し，円筒比に依存した構造形成や，流路の周長制限による影響を明らかにすることを目指した．特に，大規模計算により亜臨界遷移過程における局在乱流の確率的分裂減衰挙動を捉え，有向浸透現象（directed percolation, DP）普遍性としての特徴の有無を検討している．

1. 序論

水や空気など様々な流体の動き（流れ）は，自然・社会・産業のあらゆる場面で生じており，それらの現象予測は例えば災害軽減・感染症対策・製品向上にも直結している．近年は，流体の運動方程式を数値的に取り扱う CFD（数値流体力学）シミュレーションが如何なる問題においても活用されており，現象の理解や予測において重要なツールとなっている．計算機性能の発展も留まることを知らず，基礎研究のみならず民間航空機開発[1]や自動車後流制御[2]において 3 次元的非定常乱流場をより忠実に再現できるようになり，最適空力設計に貢献している．しかし，CFD 解析ツールもしくは流体力学そのものに残す課題は山積しており，その一つに層流—乱流間の亜臨界遷移問題がある．くだけた表現ならば，「乱流のままでいられる条件（レイノルズ数など）が分からない」のである．流体力学の教科書[3]には「円管内の流れでは Re_c（臨界レイノルズ数）≒2300」との経験則を述べており，理論的な臨界値の導出は絶望的である．また，その臨界付近で乱流は局在化して，層流と乱流が空間的に共存した流れ場になるが，その特性は未解明である．たとえ円管内という単純な流れ場であっても亜臨界遷移は未だ挑戦的な問題であり，もしも解決に至れば乱流発生・維持・減衰の高精度予測が可能となり，例えば次世代航空機開発や発電プラント安全性向上などにも資する知見獲得につながる．本研究は，規範的な流路を対象とする基礎研究ではあるが，大規模な計算領域を用いた忠実な乱流シミュレーションにより臨界状態の非常にマクロなダイナミクスを探り，亜臨界遷移にみる普遍性を明らかにするものである．本章では以降，円管内流れを含む各種壁面せん断流の亜臨界遷移に関する先行研究を概観し，本解析対象と目的について述べる．

壁面せん断流の亜臨界遷移は，前述の円管内流れで馴染み深いように，レイノルズ数 Re（例：Re = 平均流速 × 管直径 ÷ 動粘度）が線形安定性解析で求められる上臨界値 Re_L よりはるかに低いにもかかわらず，層流領域と共存しながら局在乱流または乱流塊が大規模パターンを形成する．この乱流局在化という間欠性は，主に流路形状に依存したものと考えられる．円管内流れでは管軸方向（主流方向）にのみ間欠性を有する“乱流パフ”が古くから知られ [4]，研究が続けられてきた[5]．その長い歴史から見れば，つい最近の 2011 年に Avila ら[6]が，乱流パフの分裂・減衰時間に関するレイノルズ数依存性から，大域安定性による下臨界値を見出した．さらには，

乱流パフの確率的分裂減衰挙動と有向浸透現象（directed percolation, DP）との評価がなされ[7], 1986 年の Pomeau による乱流遷移における DP 普遍性の予言[8]が実証されつつある．つまり，流体現象は Navier–Stokes 方程式による決定論的記述が可能でいながら，確率論的解釈の可能性を示唆していることが興味深い．他の流路形態においても，大規模流路実験や大型計算機による解析が活発になってきた．テイラー・クエット流 [9]，チャネル流 [10, 11]，Waleffe 流 [12]，環状クエット流 [13]，二次元コルモゴロフ流 [14]と様々なせん断流の亜臨界遷移過程に対して，DP 普遍性としての評価がなされており，間欠構造が空間 1 次元的に広がりを持つ (1+1)-D DP，または空間 2 次元的な (2+1)-D DP の特徴が報告されている．

本解析対象である環状クエット流は，流体が同軸二重円筒内で内円筒移動により軸方向に駆動された流れである（図 1）．ここでの内円筒速度は U_w で一定とし，円筒間のギャップ $h = r_{\mathrm{out}} - r_{\mathrm{in}}$ と動粘度 ν で無次元した $\mathrm{Re_w}$ が重要なコントロールパラメータである．さらに，円筒比 $\eta = r_{\mathrm{in}}/r_{\mathrm{out}}$ も重要な幾何学的パラメータとなっている．円筒比に応じて，$\eta \to 1$ で平面クエット流に漸近し，$\eta \to 0$ で極細芯を有する円管内流れとして（厳密ではないが）見立てることもできる．当該流れの亜臨界遷移域では，高円筒比で局在乱流が内円筒に螺旋状に巻き付く螺旋乱流を，低円筒比で乱流パフを形成することが知られている[13]．それぞれの間欠性は，平面クエット流および円管内流れのものに酷似しており，流路の周長制限または曲率が間欠性の空間次元に影響を及ぼしているものと考えられる．そこで，Kunii ら[13]は周方向（θ 方向）の計算領域を $L_\theta = 2\pi$ から仮想的に拡張することで，低円筒比の通常流路では見られない乱流斑点や縞パターンを検出し，これが下臨界値を引き下げると共に DP 普遍性を発現することが示された．当然ながら，この仮想拡張した環状流路は数値シミュレーションでなし得ることであり，水路実験は不可能である．しかし，周長制限を緩和した際に，方程式系が如何なる解を見せるかが力学的に興味深いところである．さらなる周領域広さの仮想拡張を行う価値はあり，大規模計算への期待がかかっている．円筒比の違いによる周長制限や曲率が間欠構造（縞〜パフ）の形成に与える影響は依然不明であり，下臨界値の円筒比依存性を含めこれら臨界現象の DP 普遍性調査が必須である．

本研究は環状クエット流亜臨界遷移の円筒比依存性と DP 普遍性の包括的な理解を目的として，円筒比 η を広い範囲で調査した．また，大規模計算機を活用して十分広い仮想周領域を設定し，可能な限り周長制限を排除した上で乱流挙動を捉えることで，臨界付近における局在乱流の維持・広がりの本質と DP 普遍性を考察する．

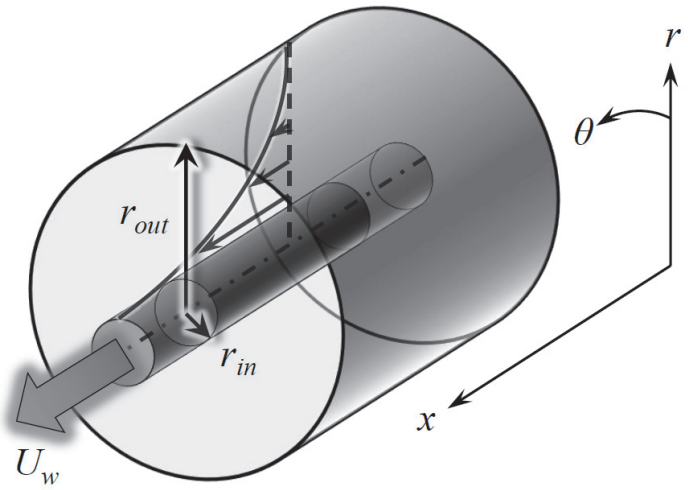


図 1　環状クエット流の模式図と座標系

2. 計算対象と方法

非圧縮性ニュートン流体の環状クエット流（図 1）を解析対象としており，x 方向に有限長の計算領域（L_x = 409.6h 〜 512h）には周期境界を課して無限の管路を表現している．主たるコントロールパラメータであるレイノルズ数 $\mathrm{Re_w}$ の減少後には十分時間をかけ，主に統計的定常状態を議論する．また，本解析対象における上臨界レイノルズ数は $\eta \geq 0.1415$ において $\mathrm{Re_L} \to \infty$で，η = 0.1 に対しても $\mathrm{Re_L} = \mathrm{O}(10^5)$である[15]．いずれの流路系に対しても $\mathrm{Re_w} \ll \mathrm{Re_L}$ のまさに亜臨界レイノルズ数領域の調査であるため，自発的に層流から乱流へ遷移することはない吸収状態転移を考えることになる．よって，一連の DNS 解析にて最初に比較的高い $\mathrm{Re_w}$ および初期擾乱を与えた場のシミュレーションを実施し，十分発達した乱流場を初期条件として準静的に $\mathrm{Re_w}$ を下げていき，大域的層流化に至るまでを各円筒比について調査した．

流体の速度 $\mathbf{u} = (u_x, u_r, u_\theta)$ と圧力 p に関する支配方程式は，円筒座標系の連続の式及び Navier–Stokes 方程式である．

$$\nabla \cdot \mathbf{u} = 0 \tag{1}$$

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \mathbf{u} \tag{2}$$

ここで，t は時間，ρ は密度である．レイノルズ数は，ギャップ半幅 h/2，円筒間相対速度の半値 U_w/2 を用いて $\mathrm{Re_w} = U_w h/4\nu$ と定義した．これは平面クエット流での定義に準拠している[16]．円筒壁面は滑り無し条件，周方向には周期境界条件を課すが，仮想的に拡張した $L_\theta > 2\pi$ の場合に対しても同様に $\theta = 0$ と L_θ の間に周期境界条件が適用される．通常流路（$L_\theta = 2\pi$）の DNS では周方向に 32〜512 分割の格子数（η が大きいほど h に対する周方向領域が拡大し，格子数も増える），仮想拡張流路（$L_\theta > 2\pi$）では 2048 の格子数を用いた．また，x 方向の格子数は 2048，r 方向には不等分割格子にて 64〜128 分割とした．時間刻み Δt は CFL 条件を満たす，可能な限り大きな値を用いている．

本解析は一切の乱流モデルを用いない直接数値解析（direct numerical simulation, DNS）を採用した．式 (1), (2) のカップリングには Fractional-step 法を用いた．時間進行は粘性項半径方向成分について 2 次精度 Crank–Nikolson 法を，その他の項に 2 次精度 Adams–Bashforth 法を用いた．空間的離散化には有限差分法を用い，主流及び周方向に 4 次精度中心差分を，半径方向に不等分割格子の 2 次精度中心差分をそれぞれ適用した．非圧縮性流体における圧力ポアソン方程式には，高速フーリエ変換で波数空間にて解く解法を採用しているが，MPI 並列計算における計算速度のボトルネックとなりやすい箇所である．

3. 結果と考察

先ず，通常環状流路（$L_\theta = 2\pi$）における間欠構造と下臨界値の円筒比依存性について述べ，本章の後半に仮想拡張した流路における流れ場の特徴について報告する．

図 2 は低円筒比および高円筒比のそれぞれに見られる典型的な乱流間欠構造である．ここでは，x 方向速度変動 u_x'の等値面を三次元可視化しており，統計的平衡状態に達した計算領域の “一部”（x 方向におよそ 1/4）を表示している．主流方向に伸びた筋状構造は高速／低速ストリークと呼ばれ，壁乱流の準秩序構造としてもよく知られるが，それが流路（壁面）全域に分布せずに偏在している．つまり，ストリーク構造が密集している箇所が局在乱流で，それ以外は準層流領域と見なすことができる．さらに観察すれば，図 2(a)は x 方向にのみ局在化した乱流塊であり円管内流れの乱流パフに酷似している．塊の長さは外円筒直径の 20 倍程度であり，これも乱流パフと同じ特徴である．以降は，図 2(a)の局在乱流を乱流パフと呼ぶ．図 2(b)は x 方向に加えて，θ 方向に

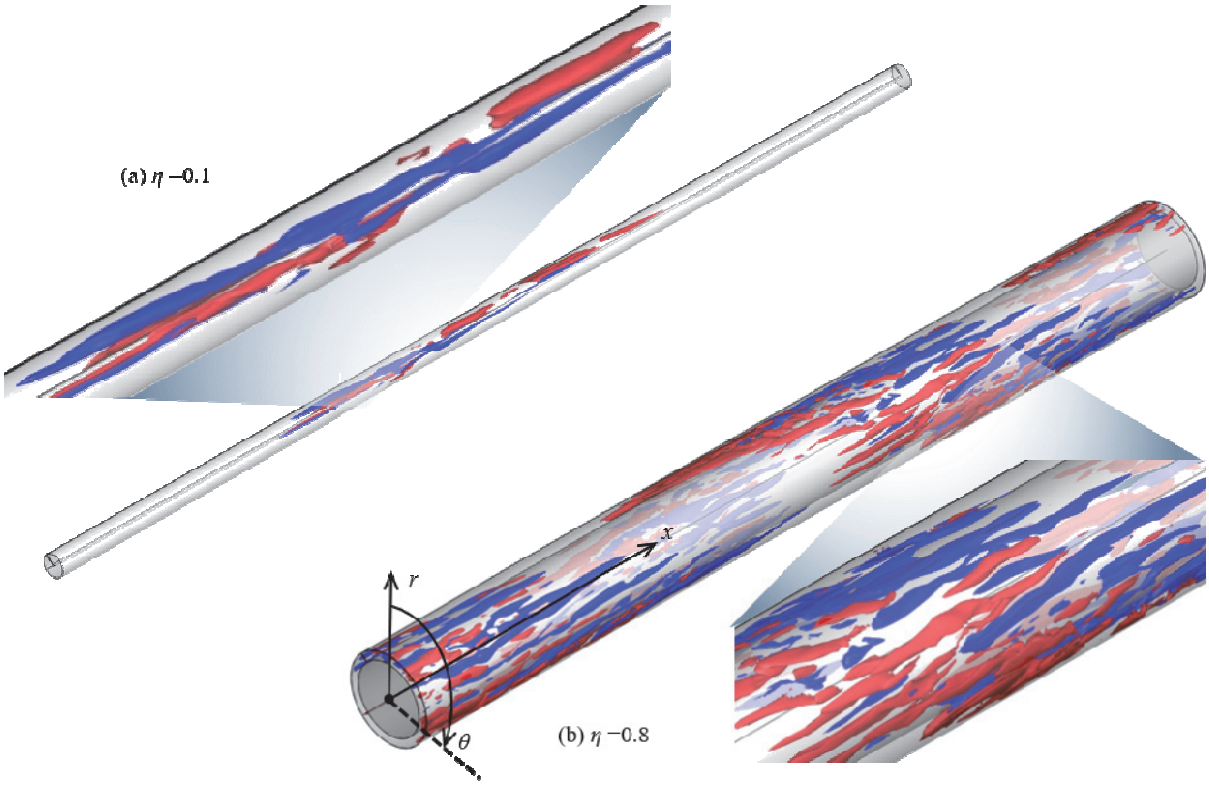


図 2　環状クエット流における乱流間欠構造．円筒比（$\eta = 0.1$，0.8）に応じて乱流パフと螺旋乱流を形成する．主流方向速度変動（赤が正の変動，青が負の変動を示す）．右上向きが主流方向で，当該方向に $\Delta x \approx 100h$ の区間を可視化．

も局在乱流の間欠性を有しており，また規則的であることは可視化結果から明白である．環状チャネル内を螺旋状に取り巻いており，これは最初に環状ポアズイユ流[17,18]で報告された螺旋乱流である．局在乱流が形成する帯は主流方向に対して一定の傾斜角度を維持しており，この角度は他の平面流で見られる乱流縞[10–12,16]とおおよそ一致している．つまり，曲率を有する流路においても周長空間がある程度まで確保されていれば平面流と同様に，さらに流れの駆動方式にもよらず，主流方向に対して斜めに帯状に局在乱流が形成されることが分かる．乱流間欠構造が乱流パフか螺旋乱流かは，円筒比 η（本質的には h に対する環状流路の周長）に依存してくる．環状ポアズイユ流においては $\eta = 0.2$〜0.4 で連続的に乱流パフ—螺旋乱流の間で遷移するが[18,19]，本研究対象の環状クエット流ではやや高い η で切替りがある[13]．本研究にて追計算によれば，$\eta = 0.5$〜0.7 の間に分岐点の存在が示唆される．ここで，図 3 に円筒間ギャップ中央の x-θ 面における半径方向速度変動 u_r'を可視化した瞬時速度場を示す．当該速度成分の空間平均値は 0 となるため，u_r 自体が速度変動成分に相当し，これの分布を可視化すれば局在乱流域と層流域の区別が明確になる．図 3 より，$\eta \geq 0.7$ では主流方向に対して傾いた帯状の局在乱流が形成される．このとき大域安定性による下臨界値 $\mathrm{Re_g}$ は同様の構造が形成される平面クエット流（$\eta \to 1$）の $\mathrm{Re_g} = 324 \pm 1$ [16]と近いになり，$\mathrm{Re_g}$ の円筒比依存性は比較的に緩やかであった．しかし，$\eta = 0.7$ の縞構造は対称性を失っており，既に構造が周長制限を受けている．$\eta = 0.5$ では大規模な二次流れを伴う縞構造が形成されず，乱流領域中に局所的な層流域が出現する“層流斑点”が形成され，明確な間欠性を表さないまま急激に層流に遷移する特徴を持つ．このときの $\mathrm{Re_g}|_{\eta=0.5}$ は $\mathrm{Re_g}|_{\eta=0.7}$ に比べて 10%ほど高くなった．また，$\eta = 0.3$ では，周方向の間欠性を失い主流方向（x 軸）にのみ間欠性を持つ乱流塊となったが，円管内流れの乱流パフに比べて x 方向に長く，異質なものが観察さ

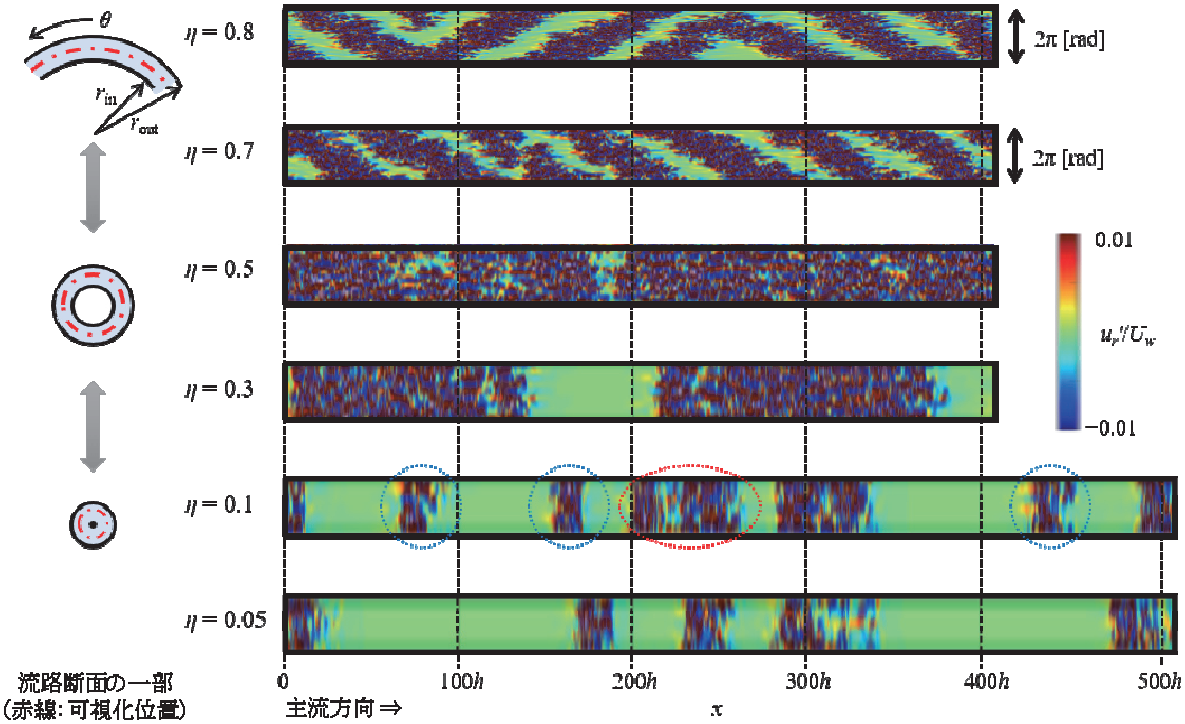


図 3　典型的な瞬時の x-θ 二次元速度分布（ギャップ中央の半径方向速度変動 u_r' 分布）．赤・青は正・負の速度を示し，緑は層流領域に相当する．縦軸は周方向座標として 2π で正規化しているため，図の縦横比は実際の面のものとは異なる．それぞれのレイノルズ数は異なるが，各円筒比における下臨界値 $\mathrm{Re_g}$ に近い条件での結果を可視化している．

れた．さらに $\eta = 0.05$〜0.1 となれば乱流パフが流路内に複数個観測されるようになった．図中の $\eta = 0.1$ の可視化図に青丸で示す箇所が乱流パフであり，赤丸で囲まれた箇所は 2 つの乱流パフに分裂する瞬間を捉えたものである．図 4 に示す乱流パフの時空間変遷は，Lemoult ら [9]のクエット流における 1 次元空間間欠性を持つ局在乱流の挙動と同様な (1+1)-D DP の確率的分裂消滅の時空間分布と定性的に一致した．図 4 には，図 3 と同様の $L_x = 512h$ での結果に加えて，$L_x = 2048h$（$N_x = 8192$）まで延長したものも示す．定性的に両結果とも同じ傾向を示しているが，図 4(b)でしばしば $500h$ 程度の間欠性（局所的な層流化）が見受けられる．さらに $\mathrm{Re_w}$ が $\mathrm{Re_g}$（≈406.9）に漸近すれば，この間欠性はさらに顕著となり，より大きな層流領域が発生し得るため，DP 普遍クラスの究明には $L_x = 2048h$ のような非常に大規模領域が依然として肝要である．Mukund & Hof [7]の実験では，円管内流れの乱流パフの変遷を長時間・長空間にわたり図示しているが，図 4 の結果ほどに乱流パフの分裂は生じておらず，さらなる検証は必要である．

周長制限を取り去るべく十分に広い周領域を持つ仮想環状流路では $\eta < 0.5$ で円筒比の低下に伴い $\mathrm{Re_g}$ は降下し，$\eta < 0.3$ で著しく降下する．前述の通常流路とは逆の傾向であることに注意されたい．図 5 に示す低円筒比における拡張流路で $\mathrm{Re_w} = 262.5 \approx \mathrm{Re_g}$ において，局在乱流が頻繁に周方向へ広がり分裂する傾向（図中 B）や周方向に局在乱流同士が対消滅するように減衰する様子（図中 A）が観察された．斑点状の局在乱流は主流方向には分裂を起こさず，主流方向に独立な周方向 1 次元的間欠性を保持する．これは，$L_\theta = 2\pi$ かつ $\eta = 0.1$ で見られた主流方向の間欠性と全く異なる点で，内円筒壁面が狭くなる低円筒比特有の強い周長制限の影響を受けていたといえる．ただし，$\mathrm{Re_g}$ から離れ，乱流斑点が主流方向に密に存在する場合，斑点は主流・周方向の 2 方向に相互作用する．つまり，$\mathrm{Re_g}$ ごく近傍以外では 2 次元空間的に広がる間欠性を持ち，$\mathrm{Re_g}$ ご

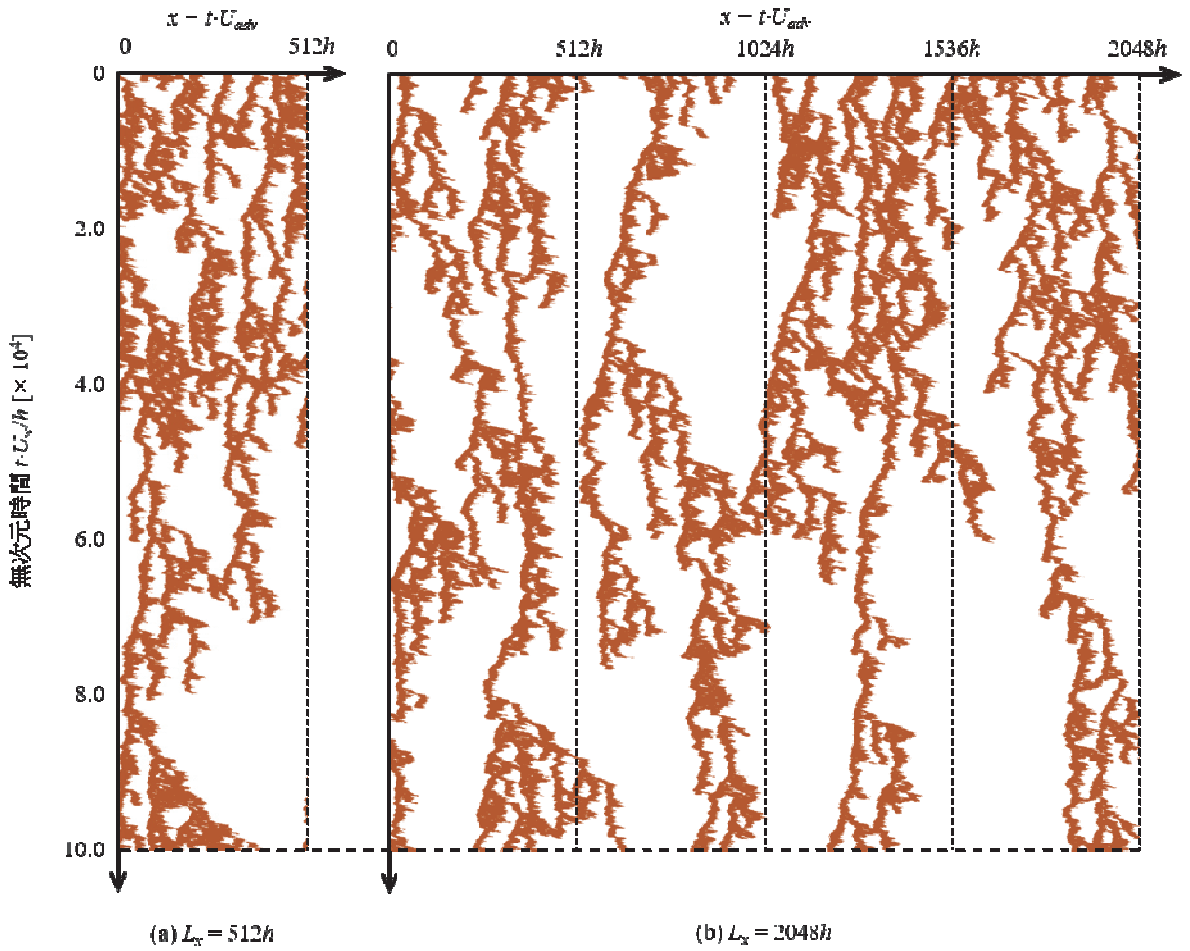


図 4　局在乱流（乱流パフ）の時空間ダイアグラム．$\eta = 0.1$ の通常環状流路（$L_\theta = 2\pi$）における $\mathrm{Re_w} = 407.5$ の結果．横軸は一定速度 U_{adv} の移動座標を示し，およそ乱流パフ伝播と一致．

く近傍で周方向にのみ広がりを持つ 1 次元の間欠性が発現する．亜臨界遷移過程と大規模間欠構造の時空間間欠性に関してスケーリングを用いて整理することで DP 普遍性の評価を行った．図 4 で示すような場を対象に，i) 臨界クエンチングによる乱流間欠率 $F_t(t)$の時間変化とそれに伴う (1+1)-D DP としてのスケーリング仮説検証, ii) 乱流間欠率の統計的定常値のレイノルズ数依存性, iii) 周方向空間間欠性に関する確率密度分布, 以上の 3 点から DP 普遍クラスとしての検証を行った．結果の詳細は既報[20]を参照されたいが，結論として(2+1)-D DP と(1+1)-D DP のクロスオーバーが同一系で実現しうることが示唆された．

4.　計算性能

本研究では，ベクトル化効率を上げた自作 Fortran コードを用いて高負荷の DNS を実施している. よって, SX シリーズなどのベクトル型計算機でその実行効率は発揮されるが, 以下に SX-ACE や次世代機の SX-Aurora TSUBASA を用いた現状性能を記す.

4.1　SX-ACE

本研究の一条件にて格子数 $N_x \times N_y \times N_z = 2048 \times 64 \times 2048$ で 100,000 ステップ分のプログラム性能を抜粋して表 1 に示す．このとき，東北大学サイバーサイエンスセンターの SX-ACE にて実行し, ノード内 OpenMP 並列化および 32 ノード間 MPI 並列化を施したコードでの検証結果である．

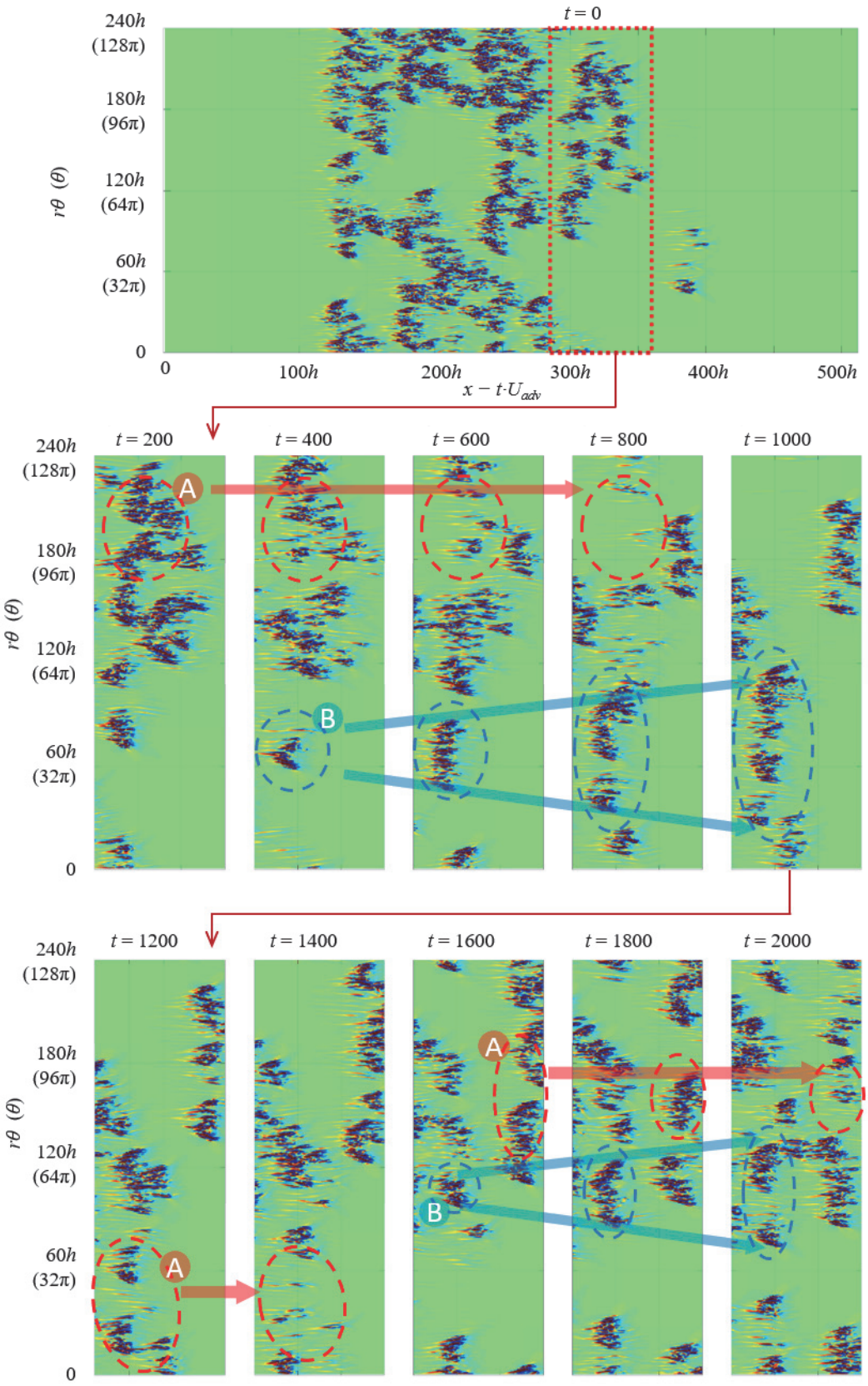


図 5　瞬時場の時系列変化．半径方向速度変動 u_r'．可視化面はギャップ中央部 $r = (r_{in} + r_{out})/2$ で，下部の図は局在乱流とおおよそ同じ速度で移動するフレームから抽出．$\eta = 0.1$ の仮想拡張流路（$L_\theta = 128\pi$）における $\mathrm{Re_w} = 262.5$ の結果．図中 A は局在乱流の対消滅が，B で分裂が見られる．

表 1　プログラム性能（SX-ACE）．100,000 ステップ分の計算結果．

実行時間 [hh:mm:ss]	14:37:38	プロセッサ同時実行時間 (1 node)	
CPU 時間 [hh:mm:ss]	1,744:57:40	Conc. Time (≥ 1) [sec]	52,109
総 Flops 値（concurrent）	419.8×10^9	Conc. Time (≥ 2) [sec]	48,079
ベクトル長	247.7	Conc. Time (≥ 3) [sec]	48,066
ベクトル換算率	99.2%	Conc. Time (≥ 4) [sec]	47,855
MIPS (concurrent)	330.6	バンクコンフリクト[sec]	24,188

4.2 SX-Aurora TSUBASA

前述の計算条件とは異なるが，同一コードによる DNS を東北大学サイバーサイエンスセンターの SX-ACE の単ノード OpenMP 並列計算と，次世代機 SX-Aurora TSUBASA を用いた同計算を比較検証した．SX-Aurora TSUBASA は当研究室が保有する A300-2 ラックモデル（1VH・2VE）を用いており，搭載 CPU は Xeon Gold 6126，ベクトルエンジンは 10B-P モデルで倍精度演算性能 2.15 TFlops のスペックである．表 2 で示されるように，単一ノードの同コア数計算で比較すれば，SX-Aurora TSUBASA は SX-ACE に比べて 4 倍近くの速度性能を有する．

表 2　プログラム性能（SX-Aurora TSUBASA）．1,000 ステップ分の 1 ノード計算結果．

使用機	コア数／ノード	格子数	実行時間
SX-Aurora	8	2048 × 64 × 1024	7.5 min
SX-Aurora	4	2048 × 64 × 1024	11.0 min
SX-Aurora	2	2048 × 64 × 1024	17.5 min
SX-Aurora	1	2048 × 64 × 1024	40.0 min
SX-Aurora	8	1024 × 64 × 1024	4.0 min
SX-Aurora	4	1024 × 64 × 1024	6.0 min
SX-Aurora	2	1024 × 64 × 1024	9.5 min
SX-Aurora	1	1024 × 64 × 1024	22.5 min
SX-ACE	4	2048 × 64 × 1024	38.5 min
SX-ACE	4	1024 × 64 × 1024	18.5 min

5. 結言

環状クエット流の亜臨界遷移を対象に DNS を実施し，間欠構造と下臨界値 Re_g の円筒比依存性を調査した．また，通常の系（$L_\theta = 2\pi$）に加え，周領域を仮想延長した系（$L_\theta \gg 2\pi$）の解析を行い，周長制限が構造形成と維持に与える影響，DP 普遍性に関して調査した．

本解析では円筒比の違いで縞ー螺旋ーパフが現れることを示した．低円筒比では周長制限により乱流パフを形成するが，下臨界値が有意に上昇した．周方向の仮想拡張により周長制限を緩和すると，局在乱流はさらに低いレイノルズ数でも維持することが分かり，特徴的な斑点状の乱流間欠構造を検出した．DP 普遍性としての特徴調査の結果，(2+1)-D と(1+1)-D DP のクロスオーバーが見られ，吸収状態（層流）直前の臨界現象は (1+1)-D DP 普遍性が顕著になることが示された．現在，円管内流れにおける乱流パフの分裂・消滅に関する理論的解釈は低次元化された力学系モデル[5]での説明もなされているが，平面流での乱流縞や高円筒比環状流での螺旋構造に対する低次元モデルは未だ存在せず，本研究のような大規模 3 次元 DNS に頼っている．今後もしばらくは，大規模 DNS の実現に SX-Aurora TSUBASA のように高性能のスーパーコンピュータ利用が不可欠ではあるが，乱流縞ー乱流パフの形成維持に関する本質的違いの説明が今後の課題であり，将来的にそれらを包括的に表現する低次元モデルの提案が期待される．

本研究結果の詳細については既報[20,21]を参照されたい．

謝辞

本研究は，東北大学サイバーサイエンスセンターのスーパーコンピュータ SX-ACE を利用することで実現することができた．また，MPI 並列計算にあたっては同センター関係各位に有益なご指導とご協力をいただき，計算結果の後処理においては Yohann Duguet 博士（仏 LIMSI-CNRS）と有意義な議論をさせて頂いた．本研究は，JSPS 科研費（基盤研究 (B) JP19H02071）の助成を受けて行われたものである．ここに記し，感謝の意を表する．

参考文献

[1] 前田一郎，“民間航空機開発における大規模 CFD 解析の適用（その 4）”，東北大学サイバーサイエンスセンター大規模科学計算システム広報 SENAC，**53(2)**，pp. 19–25, 2020.
[2] 藤井孝藏, 浅田健吾, “DBD プラズマアクチュエータを用いた自動車後流制御による抵抗低減”, 東北大学サイバーサイエンスセンター大規模科学計算システム広報 SENAC, **53(1)**, pp. 15–24, 2020.
[3] JSME テキストシリーズ 流体力学，日本機械学会，p. 93, 2005.
[4] Wygnanski, J. and Champagne, F. H., “On transition in a pipe. Part 1. The origin of puffs and slugs and the flow in a turbulent slug,” *J. Fluid Mech.*, **59**, 281–335, 1973.
[5] Barkley, D., “Theoretical perspective on the route to turbulence in a pipe,” *J. Fluid Mech.*, **803**, P1, 2016.
[6] Avila, K. *et al.*, “The onset of turbulence in pipe flow,” *Science*, **333** (2011), 192–196.
[7] Mukund, V. and Hof, B., “The critical point of the transition to turbulence in pipe flow,” *J. Fluid Mech.*, **839**, 76–94, 2018.
[8] Pomeau, Y., “Front motion, metastability and subcritical bifurcations in hydrodynamics,” *Physica D: Nonlinear Phenomena*, **23**, 3–11, 1986.
[9] Lemoult, G. *et al.*, “Directed percolation phase transition to sustained turbulence in Couette flow,” *Nature Phys.*, **12**, 254–258, 2016.
[10] Sano, M. and Tamai, K., “A universal transition to turbulence in channel flow,” *Nature Phys.*, **12**, 249–253, 2016.
[11] Shimizu, M. and Manneville, P., “Bifurcations to turbulence in transitional channel flow,” *Phys. Rev. Fluids*, **4**, 113903, 2019.
[12] Chantry, M. *et al.*, “Universal continuous transition to turbulence in a planar shear flow,” *J. Fluid Mech.*, **824**, R1, 2017.
[13] Kunii, K. *et al.*, “Laminar-turbulent coexistence in annular Couette flow,” *J. Fluid Mech.*, **879**, 579–603, 2019.
[14] Hiruta, Y. and Toh, S., “Subcritical laminar-turbulent transition as nonequilibrium phase transition in two-dimensional Kolmogorov flow,” *J. Phys. Soc. Japan*, **89**, 044402, 2020.
[15] Gittler, P. “Stability of axial Poiseuille-Couette flow between concentric cylinders,” *Acta Mech.*, **101**, 1–13, 1993.
[16] Duguet, Y. et al., “Formation of turbulent patterns near the onset of transition in plane Couette flow,” *J. Fluid Mech.*, **650**, 119–129, 2010.
[17] T. Ishida, Y. Duguet, and T. Tsukahara, Transitional structures in annular Poiseuille flow depending on radius ratio, *J. Fluid Mech.*, **794**, R2, 2016.
[18] 塚原隆裕，石田貴大，“直接数値解析による環状流路ポアズイユ流の亜臨界遷移の研究”，東北大学サイバーサイエンスセンター大規模科学計算システム広報 SENAC，**52(2)**，pp. 9–17, 2019.
[19] Ishida, T. *et al.*, “Turbulent bifurcations in intermittent shear flows: from puffs to oblique stripes,” *Phys. Rev. Fluids*, **2**, 073902, 2017.
[20] Takeda, K. *et al.*, “Intermittency and critical scaling in annular Couette flow,” *Entropy*, **22**, 988, 2020.
[21] 竹田一貴，塚原隆裕，“環状クエット流の亜臨界遷移における局在乱流構造と DP 普遍性の円筒比依存性”，日本流体力学会年会 2020 講演論文集，5 pp. (2020).

［研究成果］

野球スタジアム内気流解析に向けた数値流体計算

Numerical Fluid Simulation for Wind Analysis in BaseBall Stadium

中澤嵩[1,2]，原田昌晃[2]，尾畑伸明[2]，神原 謙悟[3]
1)大阪大学　数理・データ科学教育研究センター
2)東北大学大学院情報科学研究科　純粋・応用数学研究センター
3）株式会社楽天野球団　チーム統括本部チーム戦略室

株式会社楽天野球団が運営するプロ野球球団「東北楽天ゴールデンイーグルス」の本拠地である屋外型スタジアム「楽天生命パーク宮城（宮城球場)」(以下，野球スタジアム）では，その複雑な形状のために，風が吹き込むと複雑な気流が発生する．そして，この気流は打球の軌道に強い影響を与えることが指摘されている．本研究では，スタジアム内気流の影響を考慮することで，打球の軌道を正確に予測するソフトウェア開発を目的として，野球スタジアム内の気流解析手法の確立を目指す．このソフトウェアを活用することで，勘や経験に頼らず客観的な戦術（投手の投球，外野選手の守備陣形）を決定するための情報の一部を提供することになる．

1. はじめに

東北楽天ゴールデンイーグルスは，宮城県仙台市宮城野区にある屋外型スタジアム「楽天生命パーク宮城（宮城球場)」(以下，野球スタジアム）を本拠地としている一般社団法人日本野球機構に属するプロ野球球団である．パシフィック・リーグに所属しており，株式会社楽天野球団が運営している．

当該研究が対象とする野球スタジアムは，宮城県仙台市に位置していることから，奥羽山脈から太平洋側への北西風が年間を通して吹き下ろし，最大風速が秒速 30 メートルにも及ぶことがある（Fig. 1)．楽天野球団のチーム統括本部チーム戦略室は，この北西風が 1 塁側スタンドに吹き付けることで，野球スタジアム内で複雑な気流が発生することを把握しており，結果的に，打球に与える影響が大きいことを認識している．そこで，楽天野球団のチーム統括本部チーム戦略室は気流解析を行うことで，より正確に打球の飛距離を計算することが可能であると考えた．このような状況を鑑みて本論文では，野球スタジアム内気流の影響を考慮することで，打球の軌道を正確に予測するソフトウェア開発を目的として，野球スタジアムの位置座標の取得，位置座標の有限要素空間への埋め込み，有限要素法による数値流体計算を行ったのでその報告をする．将来的には，このソフトウェアを活用することで，勘や経験に頼らず客観的な戦術（投手の投球，外野選手の守備陣形）を決定するための情報の一部を提供することになる．

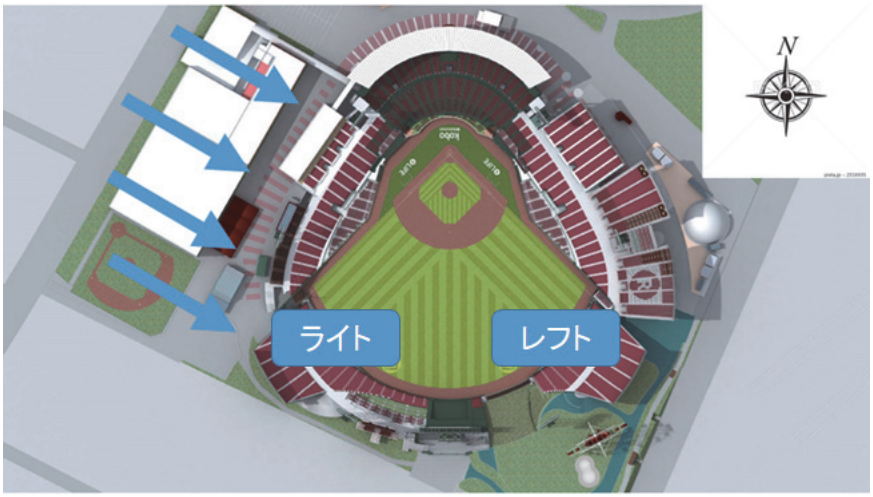


Fig. 1.　楽天生命パーク宮城（宮城球場）と北西風.

2.　航空写真画像から 3 次元点群データの生成及びスタジアム球場形状の計算領域への埋め込み

楽天生命パーク宮城（宮城球場）は，宮城野原公園宮城球場として 1950 年に開場して以来，度重なる増改築を行っている．そのため建設に必要な CAD データは複数に分散して非常に複雑な状態にあることから野球スタジアムの正確な形状を復元することは極めて困難であった．そこで，野球スタジアムの航空写真から Agisoft 社製 PhotoScan を用いることで 3 次元点群データを生成することにした．Fig. 2 は，PhotoScan の作業ウインドウであり，2 次元の航空写真から抽出した 3 次元点群データを示しており，全ての点の位置座標 $\boldsymbol{X}_{\mathrm{point}}$ は水平方向の長さが 1 の正方形となるように規格化している．Fig. 3 (a)は，ソフトウェア上で更に高精度な処理を施した後の，3 次元点群データである．

次に，数値流体シミュレーションに向けて，仮想領域法を用いて野球スタジアムの形状を計算機上で再現する．まず計算領域 Ω と境界を下記のように定義する．

$$
\begin{aligned}
&\Omega = \{\boldsymbol{x} = (x, y, z) \in \mathbb{R}^3;\ 0 \le x \le 1, 0 \le y \le 1, 0 \le z \le 0.25\},\\
&\Gamma_{\mathrm{top}} = \{\boldsymbol{x} = (x, y, z) \in \mathbb{R}^3;\ 0 \le x \le 1, 0 \le y \le 1, z = 0.25\},\\
&\Gamma_{\mathrm{bottom}} = \{\boldsymbol{x} = (x, y, z) \in \mathbb{R}^3;\ 0 \le x \le 1, 0 \le y \le 1, z = 0\},\\
&\Gamma_{\mathrm{side}} = \partial\Omega \setminus \overline{\left(\Gamma_{\mathrm{top}} \cup \Gamma_{\mathrm{bottom}}\right)}.
\end{aligned}
$$

そして，抽出した野球スタジアムの 3 次元点群データの座標 $\boldsymbol{X}_{\mathrm{point}} \subset \mathbb{R}^3$ と数値計算用に用意した有限要素空間の接点座標 $\boldsymbol{X}_{\mathrm{fem}} \subset \Omega$ とのユークリッド距離が ϵ 以下であれば $\xi(\boldsymbol{X}_{\mathrm{fem}}) = 1$，$\epsilon$ より大きい場合は $\xi(\boldsymbol{X}_{\mathrm{fem}}) = 0$ とする．

$$\xi(\boldsymbol{X}_{\mathrm{fem}}) = \begin{cases} 1 \ \ if \ \left|\boldsymbol{X}_{\mathrm{point}} - \boldsymbol{X}_{\mathrm{fem}}\right| \leq \epsilon, \\ 0 \ \ if \ \left|\boldsymbol{X}_{\mathrm{point}} - \boldsymbol{X}_{\mathrm{fem}}\right| > \epsilon. \end{cases}$$

その際，$\epsilon = 0.5 \times 10^4$ とする．それにより，有限要素法によってスタジアムの仮想的な形状を埋め込むことに成功した．実際には，数値的な安定性を向上させるために，$\xi(\boldsymbol{X}_{\mathrm{fem}})$ を外力として与えたポアソン方程式を解くことで，Fig. 3 (b)に図示するように滑らかな分布となるように再構成している．

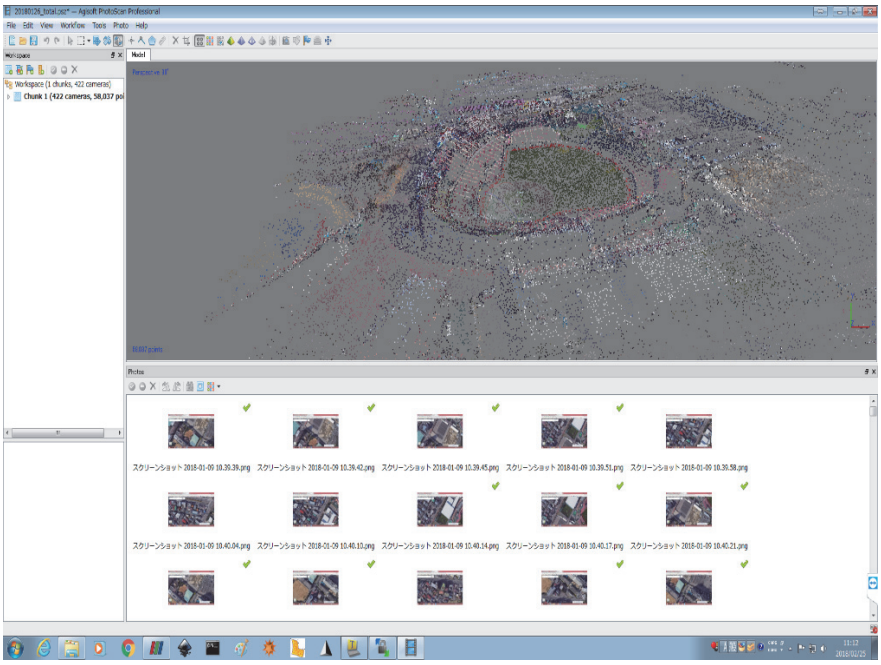
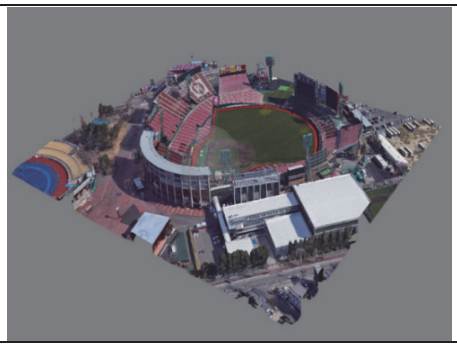
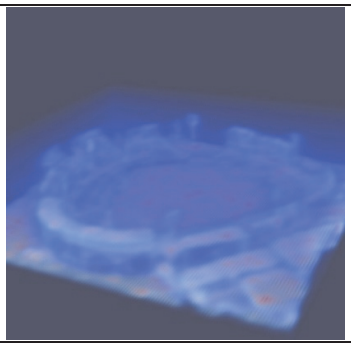


Fig. 2.　PhotoScan の作業ウインドウ．



(a)



(b)

Fig. 3.　(a)点群データと(b)有限要素空間に埋め込まれたスタジアム形状．

3.　野球スタジアム内の数値流体計算と数値計算スキーム

野球スタジアムの形状が埋め込まれた有限要素空間において，ナヴィエ・ストークス方程式の数値解析を行う．その際，速度を $\boldsymbol{u} = [u, v, w]^{\mathrm{T}}$ ，圧力を p を満たす方程式は次のようになる．

$$
\begin{aligned}
&\frac{D\boldsymbol{u}}{Dt} = -\nabla p + \frac{1}{\mathrm{Re}}\Delta\boldsymbol{u} - \alpha\xi(\boldsymbol{x})\boldsymbol{u} && \text{in } \Omega \times [0,T],\\
&\nabla\cdot\boldsymbol{u} = 0 && \text{in } \Omega \times [0,T],\\
&\boldsymbol{u} = \boldsymbol{u}^0_{\mathbb{R}^3} && \text{in } \Omega \times \{t=0\},\\
&\boldsymbol{u} = \boldsymbol{0}_{\mathbb{R}^3} && \text{on } \Gamma_{\mathrm{bottom}} \times [0,T],\\
&\boldsymbol{u} = \boldsymbol{U} && \text{on } \Gamma_{\mathrm{top}} \times [0,T],\\
&\left(p - \frac{1}{\mathrm{Re}}\nabla^{\mathrm{T}}\boldsymbol{u}\right)\boldsymbol{\nu} = \boldsymbol{0}_{\mathbb{R}^3} && \text{on } \Gamma_{\mathrm{side}} \times [0,T].
\end{aligned}
$$

ここで， Re はレイノルズ数，$\boldsymbol{\nu}$ は境界上の外向き単位法線ベクトルである．また，初期条件 $\boldsymbol{u}^0$ としてストークス方程式の解をとり，上面境界 Γ_{top} における境界条件を $\boldsymbol{U} = \left[U_x, U_y, U_z\right]^{\mathrm{T}}$ とおいた．我々の数値計算では $\mathrm{Re} = 100$ とおき， $\alpha = 10^6$ とした．更に時間積分する際には $t \in [0,T]$ として計算を行った．

4. 数値計算スキーム

数値解析を行う際には, Freefem++ [1]を用いる. 3 次元計算空間 Ω を P1-P1 要素で分割した．初期条件 $\boldsymbol{u}^0_{\mathbb{R}^3}$ はストークス方程式から得られる速度場をとる．その数値計算には Freefem++に実装されている UMFPACK [2]を用いた．ナヴィエ・ストークス方程式の計算には，時間方向に有限差分で離散化し，時間刻み $\Delta t = 0.01$ と設定し，物質微分項には特性曲線法を，粘性項には 2 次精度クランク･ニコルソン法を採用した. 更に, 乱流モデルとして LES, 安定化法として SUPG を採用した．本研究では，計算コストが膨大であるため東北大学サイバーサイエンスセンターが提供している並列コンピュータ LX406Re-2（最大 576 並列，最大演算性能 460.8GFLOPS）を 20 ノード（1 ノード 24 並列）利用した．

5. 数値計算結果

宮城県仙台市宮城野区で比較的，よく観測される北西の風及び南東の風を想定して数値計算を行った．ところで，自然風の解析ではレイノルズ数 $\mathrm{Re} \geq 10^6$ が必要であり，現実の野球スタジアムの複雑な構造を考慮すると，数値計算は極めて困難となる．そこで，自然風のもつ微細な乱流構造を解像することを目的とせず，野球スタジアム内の気流の大まかな構造を把握するために，比較的小さいレイノルズ数 $\mathrm{Re} = 100$ を採用して数値計算を行う．そのため，実際の気流をシミュレーションすることはできないが，微細な乱流構造を解像することを目的としているわけではなく，本研究では気流の大まかな構造を把握するだけで充分である．下記に具体的な数値計算結果を述べることにする．

まず初めに北西の風を計算したが，その際の上面境界条件は $\boldsymbol{U} = [1,0,0]^{\mathrm{T}}$ となり Fig. 4 (a)は U_x の空間分布である．次に，ナヴィエ・ストークス方程式を計算し，無次元化された時間 $0.1 \leq$

$t \leq 0.4$ における流れ場を Fig. 5 に図示した．この結果から，気流は 3 塁側から 1 塁側に風が吹き込み，更に 1 塁側で鉛直方向の対流が発生していることがわかった．Fig. 7 (a) に，野球スタジアムにおいて北西の風が発生した場合における鉛直方向の対流が発生するエリアを赤枠で示している．

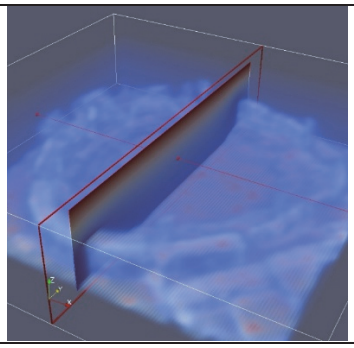
次に，南東の風を想定した数値シミュレーションを上面境界条件は $\boldsymbol{U} = [-1,0,0]^{\mathrm{T}}$ のもとで，ナヴィエ・ストークス方程式を計算した．Fig. 4 (b)は U_x の空間分布であり，Fig. 6 は無次元時間 $0.2 \leq t \leq 0.8$ における流れ場を Fig. 6 に図示した．その結果から，北西の風と逆向きの気流が発生するとともに，3 塁側に鉛直方向の対流が発生していることがわかった．Fig. 7 (b) に，南東の風が発生した場合における鉛直方向の対流が発生するエリアを赤枠で示した．

6. 考察

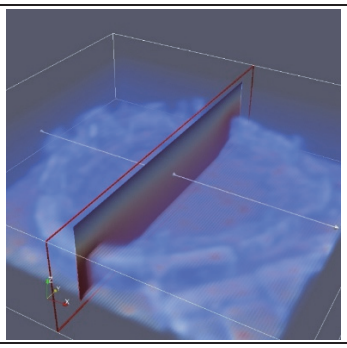
本研究では，株式会社楽天野球団が運営するプロ野球球団「東北楽天ゴールデンイーグルス」の本拠地である屋外型スタジアム「楽天生命パーク宮城（宮城球場）」において，打球の軌道を正確に把握することを目的として野球スタジアム内の気流解析手法を構築した．具体的には，まず初めに Agisoft 社製 PhotoScan を用いて野球スタジアムの位置座標の取得，次に位置座標の有限要素空間への埋め込み，そして最後に有限要素法による数値流体計算を行った．

ところで本研究では，微細な乱流構造を解像することを目的としている訳ではなく，野球スタジアム内の大まかな気流構造を把握することが重要であることから，比較的小さいレイノルズ数 $\mathrm{Re} = 100$ を採用した．その上で，宮城県では比較的，よく観測される北西の風及び南東の風を想定して数値計算を行った．数値シミュレーションの結果，北西の風の場合には，気流は 3 塁側から 1 塁側に風が吹き込み, 更に 1 塁側に鉛直方向の対流が発生していることがわかった. 一方で，南東の風の場合，北西の風と逆向きの気流が発生するとともに，3 塁側に鉛直方向の対流が発生していることがわかった．

このシミュレーションとは別に，実際に公式戦が行われた日に野球スタジアムに風向風速計を設置して計測を行っている．事前の風向の実測により，野球スタジアム内気流を強く反映していると判断出来る 4 箇所（1 塁側上段， 1 塁側内野席の中間点，ライトのポール後方，バックスクリーン横）に，風向風速計を設置した（Fig. 8）．現時点では，野球スタジアムの形状については実際に近い状態を再現している一方，レイノルズ数については $\mathrm{Re} = 10^6$ を想定すべきであるが，数値計算を安定に行うために $\mathrm{Re} = 10^2$ を採用している．そのため，シミュレーションの結果と実測との整合性などの検証は出来ていないが，研究を展開する際にはこの計測結果が役立つと思われる．将来的には，打球の風による影響を正確に予測するソフトウェアを開発し，勘や経験に頼らず客観的な戦術（投手の投球，外野選手の守備陣形）を決定するための情報の一部を提供するソフトウェアの開発を目指す．

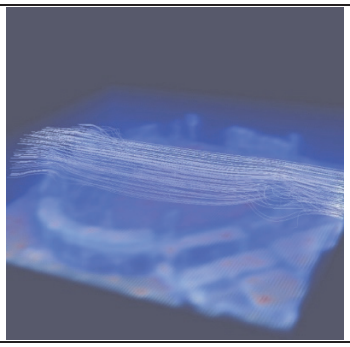


(a) $U_x = 1$

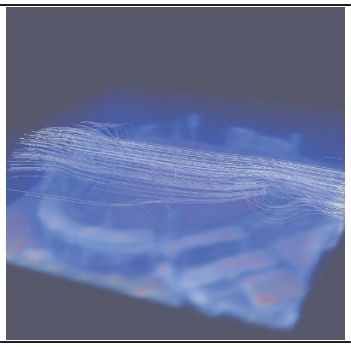


(b) $U_x = -1$

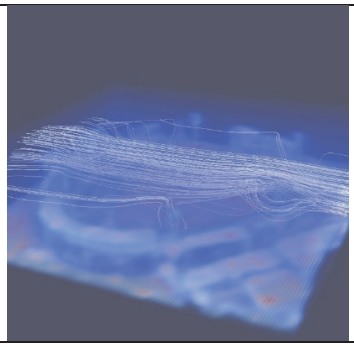
Fig. 4.　(a)北東の風，(b)南東の風.



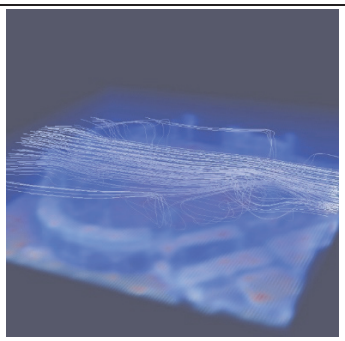
$t = 0.1$（10 step）



$t = 0.2$（20 step）

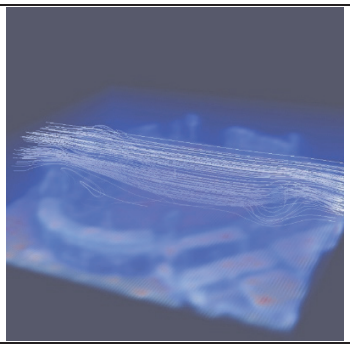


$t = 0.3$（30 step）

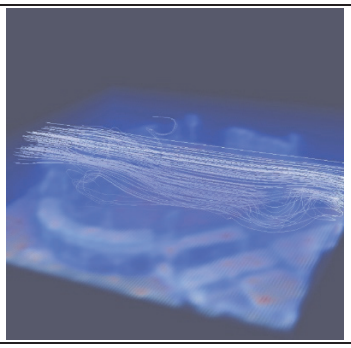


$t = 0.4$（40 step）

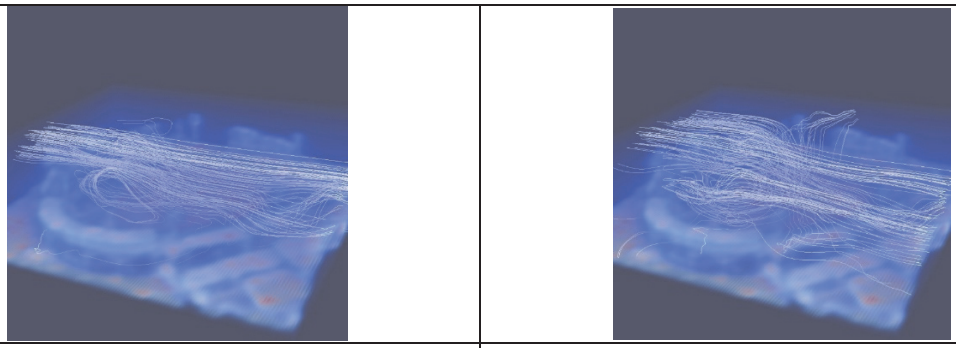
Fig. 5.　北西の風における流れ場.



$t = 0.2$（20 step）

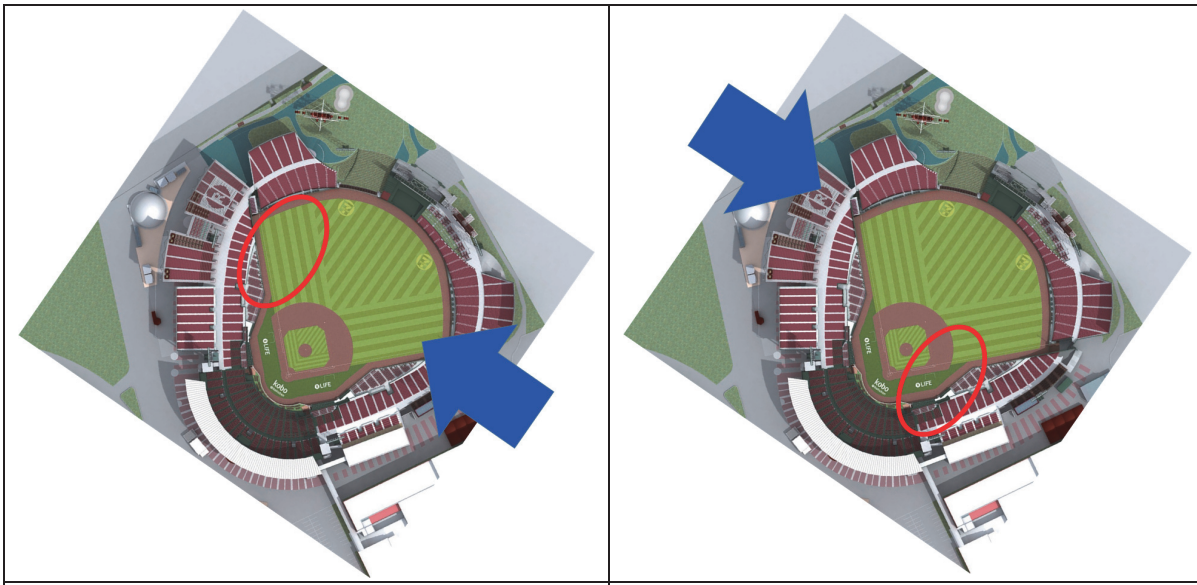


$t = 0.4$（40 step）



$t = 0.6$（60 step）	$t = 0.8$（80 step）

Fig. 6.　南東の風における流れ場.



(a) 北西の風	(b) 南東の風

Fig. 7.　楽天生命パーク宮城（宮城球場）において，(a) 北西の風と (b) 南東の風が発生した場合における鉛直方向の対流が発生するエリアを赤枠で示す.

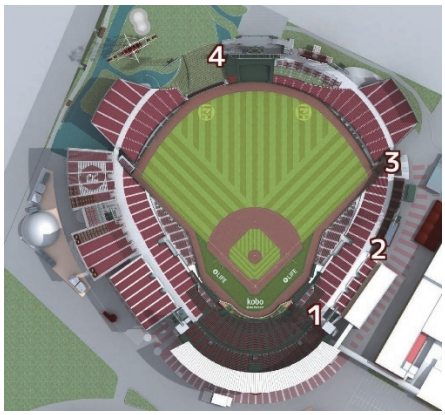


Fig. 8.　楽天生命パーク宮城（宮城球場）において，風向風速計を設置した箇所．１：１塁側上段，２：１塁側内野席の中間点，３：ライトのポール後方，４：バックスクリーン横

謝辞

本研究は，東北大学大学院情報科学研究科における研究科重点プロジェクト「数学と諸分野の協働推進による学際的･総合的な新領域研究開拓」（2016年度～2018年度）による支援を受けた．更に，本研究は，東北大学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した．

参考文献

[1] Hecht, F., New development in FreeFem++, J. Numer. Math. 20, pp. 251–265, 2012

[2] Davis, Timothy A., Algorithm 832. ACM Transactions on Mathematical Software, 30 (2), pp. 196–199, 2004.

［報告］

JHPCN 学際大規模情報基盤共同利用・共同研究拠点 第 12 回シンポジウム報告

滝沢 寛之
東北大学サイバーサイエンスセンター スーパーコンピューティング研究部

第 12 回学際大規模情報基盤共同利用・共同研究拠点(JHPCN)シンポジウムは，令和２年 7 月 9，10 日に東京品川 The Grand Hall での開催が予定されていましたが，新型コロナウイルス(COVID-19)感染拡大の懸念があることからオンライン開催となりました．今年度のシンポジウムでは昨年度採択された 58 課題のうち，13 課題に関するオンライン講演と 43 課題のオンラインポスター発表がありました．また，今年度採択された課題 52 件のオンラインポスター発表に加えて，各構成拠点から推薦された萌芽課題 9 件のポスター発表，および 2021 年度提供予定の計算資源の紹介がありました．例年とは異なり Slack を用いた質疑応答とはなりましたが，活発な議論が展開されました．

学際大規模情報基盤共同利用・共同研究は，東北大学，北海道大学，東京大学，東京工業大学，名古屋大学，京都大学，大阪大学，九州大学にそれぞれ附置するスーパーコンピュータを持つ 8 つの共同利用の施設を構成拠点とする「学際大規模情報基盤共同利用・共同研究拠点」が，超大規模数値計算系応用分野，超大規模データ処理系応用分野，超大容量ネットワーク技術分野，およびこれらの技術分野を統合した大規模情報システム関連研究分野，更には分野間に亘る複合分野の研究者らと取り組む学際的な共同利用・共同研究です．

平成 22 年度から令和 2 年度の 9 年間において約 400 件を超える課題が学際大規模情報基盤共同利用・共同研究として採択されており，そのうち当センターとの共同研究課題は約 60 件にとなっております．今年度は，7 件が当センターとの共同研究課題として採択されております．今年度採択分も含め，これまでの採択課題に関する情報は以下の URL で公開されております。是非，高性能計算を用いた多岐にわたる共同研究活動をご覧頂ければと存じます(JHPCN URL: https://jhpcn-kyoten.itc.u-tokyo.ac.jp/ja/sympo/12th)．

来年度の学際大規模情報基盤共同利用・共同研究拠点の研究公募は 11 月頃に公開予定です．ご興味のあるかたは奮ってご応募ください．また，応募に際しまして当センターの計算機科学を専門とする教員との共同研究の可能性を検討したい，手続き方法が分からない等，本応募に関して不明な点があります場合は，お気軽に当センターまでお問い合わせください．

【JHPCN に関する問い合わせ窓口】joint_research@cc.tohoku.ac.jp

[Web 版スーパーコンピュータ AOBA のお知らせより]

スーパーコンピュータ AOBA のお知らせに掲載された記事の一部を転載しています。

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

新スーパーコンピュータシステムの愛称決定

東北大学サイバーサイエンスセンターは、2020 年 10 月から運用を開始する新スーパーコンピュータシステムの愛称を、AOBA に決定しました。また、愛称に合わせロゴマークも決定しました。

全国共同利用型のスパコンとして、皆さんに親しまれ、多くの人に利用していただけるよう、また学術研究から社会基盤まで多様な研究・開発を支えるスパコンの機能や魅力を、青々とした葉が盛んに伸びゆく新緑の青葉(AOBA) のように、青葉(AOBA)山の地から世界に向けて、スーパーコンピュータ AOBA と共に発信していきます。

愛称　　：　スーパーコンピュータ AOBA
ロゴマーク：

愛称は 7 月 1 日から 31 日まで 1 か月間の一般公募を行い、総数 346 件のご応募を頂きました。なお、採用愛称には複数のご応募がありましたので、厳正なる抽選の結果、茨城県在住の飯村様（専業主婦）を最優秀賞の受賞者と決定させていただきました。たくさんのご応募をいただきまして、誠にありがとうございました。

［お問い合わせ先］
東北大学サイバーサイエンスセンター
E-mail：cc-scg@grp.tohoku.ac.jp

（スーパーコンピューティング研究部，共同利用支援係，共同研究支援係）

— SENAC 執筆要項 —

１．お寄せいただきたい投稿内容

サイバーサイエンスセンターでは、研究者・技術者・学生等の方々からの原稿を募集しております。以下の内容で募集しておりますので、皆さまのご投稿をお待ちしております。なお、一般投稿いただいた方には、謝礼として負担金の一部を免除いたします。

・一般利用者の方々が関心をもたれる事項に関する論説
・センターの計算機を利用して行った研究論文の概要
・プログラミングの実例と解説
・センターに対する意見、要望
・利用者相互の情報交換

２．執筆にあたってご注意いただく事項

(1)原稿は横書きです。
(2)術語以外は、「常用漢字」を用い、かなは「現代かなづかい」を用いるものとします。
(3)学術あるいは技術に関する原稿の場合、200 字～400 字程度のアブストラクトをつけてください。
(4)参考文献は通し番号を付し末尾に一括記載し、本文中の該当箇所に引用番号を記入ください。
　・雑誌：著者, タイトル, 雑誌名, 巻, 号, ページ, 発行年
　・書籍：著者, 書名, ページ, 発行所, 発行年

３．原稿の提出方法

原稿のファイル形式は Word を標準としますが、PDF での提出も可能です。サイズ*は以下を参照してください。ファイルは電子メールで提出してください。

－Word の場合－
　・用紙サイズ：A4
　・余白：上＝30mm　下＝25mm　左右＝25mm　綴じ代＝0
　・標準の文字数（45 文字 47 行）

＜文字サイズ等の目安＞
　・表題＝ゴシック体 14pt 中央　・副題＝明朝体 12pt 中央
　・氏名＝明朝体 10.5pt 中央
　・所属＝明朝体 10.5pt 中央
　・本文＝明朝体 10.5pt
　・章・見出し番号＝ゴシック体 11pt～12pt
　　*余白サイズ、文字数、文字サイズは目安とお考えください。

４．その他

(1)執筆者には希望により本誌 PDF 版を進呈します。
(2)一般投稿を頂いた方には謝礼として、負担金の一部を免除いたします。免除額は概ね 1 ページ 1 万円を目安とします。詳細は共同利用支援係までお問い合わせください。
(3)投稿予定の原稿が 15 ページを超す場合は共同利用支援係まで前もってご連絡ください。
(4)初回の校正は、執筆者が行って、誤植の防止をはかるものとします。
(5)原稿の提出先は次のとおりです。

東北大学サイバーサイエンスセンター内　情報部情報基盤課共同利用支援係
e-mail　uketuke@cc.tohoku.ac.jp
TEL　　022-795-3406

スタッフ便り

社会のデジタル化に最も貢献したのは IT でも政府でもなく COVID-19 であるといった具合に、一時期、国内外を問わず揶揄されていたのですが、最近はデジタル庁が置かれるなど国内でも急速に ICT 対応の推進が始まったように見えます。同時に、誰かが ICT 化を阻害していたとか、現場向けの技術開発ができていないとか、そもそも必要な所に経済的・人的リソースが割かれていないとか、様々な問題・課題が見えてきたようです。このうち、阻害している「誰か」というのが、力を持った少人数の場合もあれば、しばしば正当な理由なき理不尽な拘りを持った不特定多数であったりするところに、無責任さを感じざるを得ないことがあります。

何か新しいことを取り入れればよいというわけでは勿論なく、見極めは重要なのですが、自らが期せずして正当な発展を止める側に立っていたりしないかどうか、新しいものを作り出す大学・学校なら特に、気を付けておく必要があるでしょう。国際的に遅れが著しいというデータもある、生徒の ICT 環境とリテラシですが、これらを正していくために、適切な所に適切な人員の配置が望まれます。(後藤英昭)

「SENAC」のスタッフ便りを書くことになった。ところで、「SENAC」って何かの略かなあ？調べてみたら、日本電気㈱と共同開発し、昭和 33 年に公開された当時最大級の電子計算機 SENAC-1(SENdai Automatic Compuer)から名付けたとのことであった。その後昭和 61 年には初代のスーパーコンピュータが導入され、今年 10 月から 8 代目が稼働を開始した。「合計 576 基の VE を搭載する「SX-Aurora TSUBASA」、総理論演算性能約 1.77PFLOPS」と言われても何がなんだか、「すごいんだろうなあ」以外思い浮かばない。当センターの展示室には、SENAC-1 や歴代スパコンのユニット、写真などが展示され見ることができますよ。

私がコンピュータを始めて触った当時をちょっと思い出してみた。「N88」: 最初に出会ったパソコンだった。「フロッピーディスク(FD)」: 128KB/8 インチでふにゃふにゃケースだったのが、1.2MB/3.5 インチのハードケースに進化し感動したなあ。「音響カプラ」:「ゴー、キーキー」あの音が如何にも頑張って通信しているぞって聞こえた。ああ、懐かしい。もう一度 N88 とカプラを使ってメールを送ってみたくなった・・・昔をふりかえるとは、年だなあ。(K.M)

SENAC 編集部会
滝沢寛之　水木敬明　後藤英昭　伊藤昭彦
早坂和勝　大泉健治　小野　敏　斉藤くみ子

令和 2 年 10 月発行
編集・発行　東北大学
サイバーサイエンスセンター
仙台市青葉区荒巻字青葉 6-3
郵便番号　980-8578
印　刷　株式会社　東誠社

スーパーコンピュータ AOBA システム一覧

計算機システム	機　種
サブシステム AOBA-A	SX-Aurora TSUBASA
サブシステム AOBA-B	LX 406Rz-2

サーバとホスト名

ログインサーバ	login.cc.tohoku.ac.jp
データ転送サーバ	file.cc.tohoku.ac.jp

サービス時間

利用システム名等	利用時間帯
サブシステム AOBA-A	連　続　運　転
サブシステム AOBA-B	連　続　運　転
サ ー バ	連　続　運　転
館内利用	平日 8:30～21:00

サブシステム AOBA-A の利用形態と制限値

<table>
<tr><th>利用形態</th><th>キュー名</th><th>VE 数※</th><th>実行形態</th><th>最大経過時間
既定値/最大値</th><th>メモリサイズ</th></tr>
<tr><td>無料</td><td>sxf</td><td>1</td><td>1VE</td><td>1 時間/1 時間</td><td rowspan="5">48GB×VE 数</td></tr>
<tr><td rowspan="3">共有</td><td rowspan="2">sx</td><td>1</td><td>1VE</td><td rowspan="3">72 時間/720 時間</td></tr>
<tr><td>2～256</td><td>8VE 単位で確保
（VH を共用しない）</td></tr>
<tr><td>sxmix</td><td>2～8</td><td>1VE 単位で確保
（VH を共用する）</td></tr>
<tr><td>占有</td><td colspan="4">個別設定</td></tr>
</table>

※2VE以上を利用した並列実行にはMPIの利用が必用

サブシステム AOBA-B の利用形態と制限値

<table>
<tr><th>利用形態</th><th>キュー名</th><th>ノード数※</th><th>最大経過時間
既定値/最大値</th><th>メモリサイズ</th></tr>
<tr><td>共有</td><td>lx</td><td>1～16</td><td>72 時間/720 時間</td><td rowspan="2">256GB×ノード数</td></tr>
<tr><td>占有</td><td colspan="3">個別設定</td></tr>
</table>

※2ノード以上を利用した並列実行にはMPIの利用が必用

目次

東北大学サイバーサイエンスセンター
大規模科学計算システム広報 Vol.53 No.4 2020−10

［大規模科学計算システム］
SX-Aurora TSUBASA のハードウェア ……星　宗王　1
江副　健司
上山根　慎

LX 406Rz-2 のハードウェア ……横山　淳　10
乾　重人
水子　陽一

利用申請と利用負担金について ……16

鍵ペアの作成とログイン方法 ……26

サブシステム AOBA-A の利用法 ……33

サブシステム AOBA-B の利用法 ……45

バッチリクエストによるジョブの実行方法 ……53

ストレージシステムの利用法 ……62

SX-Aurora TSUBASA でのプログラミング（ベクトル化編）……岡野　進一　68
工藤　淑裕

［共同研究成果］
複数領域 FDTD（時間領域差分）法の高速化 ……有馬　卓司　85
宇野　亨

直接数値解析を用いた仮想拡張した環状流路における亜臨界遷移の研究
……塚原　隆裕　90
竹田　一貴

［研究成果］
野球スタジアム内気流解析に向けた数値流体計算 ……中澤　崇　99
原田　昌晃
尾畑　伸明
神原　謙悟

［報　告］
JHPCN 学際大規模情報基盤共同利用・共同研究拠点第 12 回シンポジウム報告 …… 107

［Web 版大規模科学計算システムニュースより］
新スーパーコンピュータシステムの愛称決定 …… 108

執筆要項 …… 109

スタッフ便り …… 110

VEGETABLE OIL INK