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.57 No.2 2024−4

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デジタルサービス支援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) 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 9:00～21:00
	利用者談話室	(3444)	自販機	8:30～21:00
	展 示 室*（分散コンピュータ博物館)*	*見学希望の方は受付までご連絡ください。	歴代の大型計算機等の展示	9:00～16:00
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
	スーパーコンピューティングサポートユニット	[受付] 022-795-3406(3406) cc-uketuke@grp.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
		022-795-6252(6252) cc-sys@grp.tohoku.ac.jp	共同研究、計算機システムに関すること	8:30～17:15
	デジタルプラットフォームユニット	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
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)	

＊（　）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。

[共同研究成果]

日本域領域再解析 RRJ-Conv における線状降水帯の統計解析

伊藤純至、松島沙苗（東北大学理学研究科）
福井真、 廣川康隆（気象研究所）

日本領域再解析 RRJ-Conv の計算を進めており、2021 年から遡り過去 40 年間のデータセットを整備した。そのうち観測結果を基にした解析雨量と比較可能な過去 30 年分のデータセットより線状降水帯を抽出し、経年変化を解析した。RRJ-Conv より抽出された線状降水帯の個数は、解析雨量と同程度であったが、合致したものは２割程度であった。過去 30 年間では観測より緩いペースではあるものの、増加トレンドがみられている。

1. はじめに

線状降水帯の特徴や経年変化の理解のため、気象レーダー観測と地上雨量観測（アメダス）を合成した解析雨量を用いた線状降水帯の抽出と統計解析が行われている。しかし、観測網や解析手法の変遷の影響を受ける解析雨量をもとに長期変化を調べることは容易でない。東北大学と気象研究所の共同研究により、日本域を水平格子間隔 5km で計算する領域再解析のデータセット(RRJ-Conv)[1]が作成された。RRJ-Conv は従来型データのみを同化することによって長期間の一貫性を維持しつつ、全球再解析では解像できない線状降水帯のようなメソスケールの極端気象の再現が期待される。そのため線状降水帯の経年変化の検証等に有用と考えられる。

本研究では、解析雨量と比較しながら、RRJ-Conv において再現される線状降水帯の特徴を確認した上で、線状降水帯の発生数の経年変化を調べる。

2. 使用データ・手法

使用データは、水平格子間隔 5km の解析雨量および RRJ-Conv の 1 時間積算降水量で、対象期間は 1991 年 1 月から 2020 年 12 月の 30 年間である。

線状降水帯の抽出には Hirockawa et al.(2020)[2]と同一の手法・基準を用いた。この手法では、まず３時間積算降水量を用いて 80㎜/3h の等値線に囲まれ、最大値が 100 ㎜/3h 以上となる強雨域を抽出する。抽出した強雨域を時空間の連続性や形状の特徴をもとに線状降水帯であるかどうかの判定をしている。

解析雨量で抽出された線状降水帯の開始時刻および終了時刻の前後 5 時間以内かつ最大積算降水量の格子が半径 200km 以内に、RRJ-Conv で線状降水帯が抽出された場合を一致事例として判定した。

3. 結果

3.1 RRJ-Conv の線状降水帯の再現の確認

解析領域を Hirockawa et al.(2020)[2]と同一の日本付近とした場合、1991 年～2020 年の 30 年間において RRJ-Conv で抽出された線状降水帯は 717 事例となった。なお解析雨量においては同期間に 747 事例抽出されている。そのうち RRJ-Conv と解析雨量が抽出した事例が一致したものは 147 事例であった。つまり、RRJ-Conv と解析雨量の検出事例数はほぼ同等であるが、一致数は 2 割程度となった。解析雨量及び RRJ-Conv から抽出された線状降水帯の分布を比較したところ、九州を中心とした西日本太平洋側での出現数が圧倒的に多い特徴はよく一致していた。RRJ-Conv の線状降水帯の抽出位置を解析雨量と比較すると陸上では少なく、海上で多くなる傾向であった。

3.2 線状降水帯出現頻度の経年変化

抽出した線状降水帯の年別出現頻度を図 1 に示す。10 年間で約 3 事例のペースの増加がみ

られた。解析雨量からの抽出数の経年変化と比較すると、ある程度の相関がみられる一方、RRJ-Conv の方が増加のペースは緩くなっている。これは解析雨量の抽出数と比較して、RRJ-Conv の抽出数が統計期間の前半(1991 年～2005 年)は多く、後半(2006 年～2020 年)は少ない傾向となったためである。一方、統計期間の後半(2006 年～2020 年)において北日本での事例数が増加している等の特徴は解析雨量の結果と一致していた。

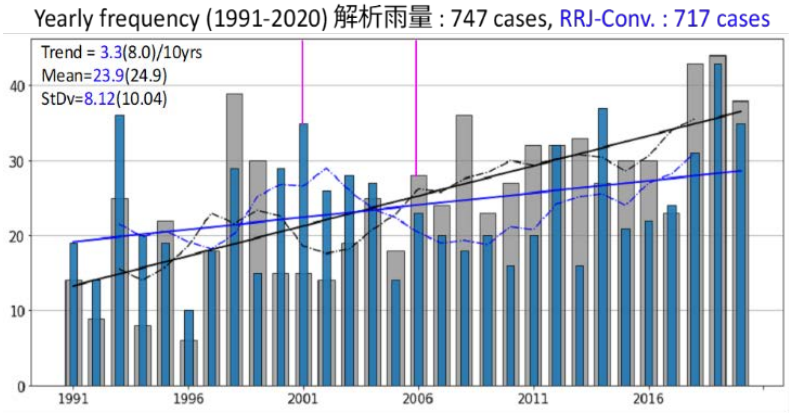


図１　1991 年～2020 年に抽出された線状降水帯の解析雨量(黒)と RRJ-Conv(青)における年別出現頻度。実線は長期変化傾向、破線は５年移動平均を示す。赤縦線は解析雨量の水平解像度 5→2.5→1km の変更時期を示す。

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。本研究は、東北大学と気象庁気象研究所の共同研究の一環として行った。また、本研究は JST 共創の場形成支援プログラム JPMJPF2013 および科研費 22H01295 の支援を受けた。

参考文献

[1] Fukui, S., T. Iwasaki, K. Saito, H. Seko, and M. Kunii, A feasibility study on the high-resolution regional reanalysis over Japan assimilating only conventional observations as an alternative to the dynamical downscaling. *J. Meteor. Soc. Japan,* **96**, 565–585, 2018.
[2] Hirockawa, Y., T. Kato, H. Tsuguti, and N. Seino, Identification and classification of heavy rainfall areas and their characteristic features in Japan. *J. Meteorol. Soc. Japan,* **98,** 835-857, 2020.

［共同研究成果］

有限温度におけるプロトン伝導体のドーパントの配置

藤﨑　貴也　島根大学　材料エネルギー学部

本研究では、水蒸気電解と燃料電池の電解質として利用可能なプロトン伝導性酸化物に焦点を当てた。それはペロブスカイト構造に三価のカチオンを添加したもので、有限温度下でのカチオンの位置について密度汎関数理論とクラスター展開法を用いて調査した。その結果、カチオンは800℃でクラスターを形成せずに孤立していることが示された。

1　はじめに

化石燃料の減少に伴い、再生可能エネルギーと水素の組み合わせによる水素社会がますます注目を集めている。水素はその大規模な電力貯蔵の特性から、持続可能なエネルギー源としての期待が高まっている。特に、水素社会の発展において鍵となるのが燃料電池であり、その電解質としてプロトン伝導性酸化物（プロトン伝導体）の応用が期待されている。この応用により、より効率的で持続可能な燃料電池の開発が可能となる。プロトン伝導体は、ペロブスカイト構造（組成式：ABO_3）の B サイトに 3 価のカチオンを一部導入することで結晶が電気的中性条件を保つ特徴を持っている。ここでの 3 価のカチオンはドーパントと呼ばれ、図 1 では M 原子を指す。この特殊な構造により、結晶内に酸素空孔が形成され、これに水分子が結合することで水和が起こる。この過程において、水分子の水素原子が酸素空孔にプロトンを供給し、その結果としてプロトン伝導が発生する仕組みである。この研究や技術の進展は、将来的なエネルギーの持続可能性や効率向上に向けた一歩となる。特に、プロトン伝導体を用いた燃料電池技術は、クリーンで効率的なエネルギー供給の実現に寄与することが期待されている。

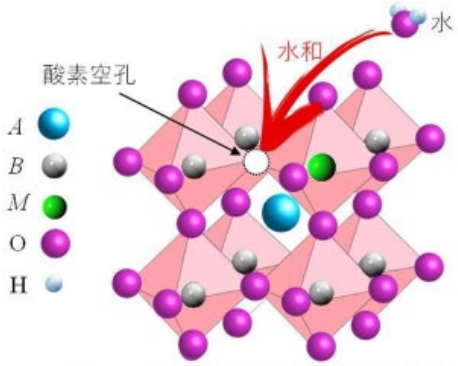


図1　プロトン伝導体の結晶構造と水和の模式図

これまで、プロトン伝導体の水和による体積膨張（化学膨張）の評価は、理論計算と実験的手法を駆使して行われてきたが、その中で注目すべきなのはドーパントの配置に依存した格子エネルギーが最小でない場合の計算結果が実験事実と一致したことである[1]。これに関して、具体的な実験データを挙げると、固相反応法によって調製されたプロトン伝導体の中で、Y を添加したストロンチウムセレートが、同じく Y を添加したストロンチウムジルコネートよりも水和による体積膨張が著しく大きかったことが確認された。この実験事実に基づいて行った理論計算では、セレートとジルコネートにおける一つの水分子がもたらす影響を詳細に調査した。

2　解析手法と結果

本研究では、第一原理計算と有限温度における結晶の配置を算出できるクラスター展開を用いた。その結果、Y 原子が最も遠い位置にある状態が、プロトン伝導性酸化物の実験の傾向と驚くほど一致していることが明らかとなった（図 2)。ただし、この Y 原子の持つ結晶構造は、理論計算によれば格子エネルギーが最小ではなく、従って最も安定な結晶構造ではないことが判明した。

この研究結果は、従来の格子エネルギーに基づくアプローチが物質の挙動を正確に捉えることができない場合があることを示唆しており、将来の研究においてはより複雑な要因を考慮に入れた理論モデルの開発が求められることを示唆している。

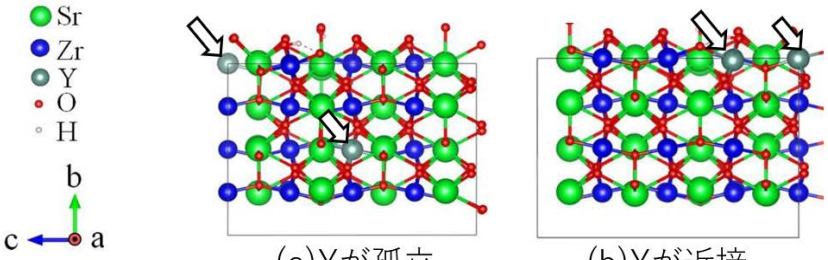


図2 YをSrZrO3とSrCeO3に対してクラスター展開によって計算された構造。Yが孤立している場合は0 K、Yが孤立していないときは1073 Kを示す。

3　考察

本研究によって、第一原理計算とクラスター展開法の組み合わせによって有限温度における Y 原子の配置を予測することが出来た。しかしながら実験的にこれを明らかにすることは非常に困難である。しかしながら、イオンミーリング等で非常に薄いサンプルを作り、透過型電子顕微鏡と EDX 解析は二次元方向しかマッピングによってY原子の配置をある程度は特定することが出来るかもしれないため、今後はそのような実験的アプローチが必要になると思われる。

4　結論

本研究の実施により第一原理計算とクラスター展開法がプロトン伝導性酸化物の有限温度の原子配置を予想することが可能であることが明らかになった。本研究は実験的に裏付けが望まれるが、そのための大きな足掛かりを築けたと考えられる。

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。

参考文献

[1] Fujisaki, T et.al.,(2019) Solid State Ionics 333, 1–8.
[2] Ki m, N. et . al . , (2019) Chemi st ry of Mat eri al s, 31(1) , 233–243.

［共同研究成果］

プラズマアクチュエータによる角部剥離流れ制御の性能向上に向けて

浅田　健吾：東京理科大学工学部情報工学科
渡部　航太朗：東京理科大学工学部情報工学科
関本　諭志：東京農工大学工学部機械システム工学科
藤井　孝藏：東京理科大学工学部情報工学科

1.　はじめに

流れの制御は古くから流体研究分野では流体機器の性能を向上するための重要なテーマとされてきた．これまで様々な流体制御手法が考案されてきたが，近年では DBD (Dielectric Barrier Discharge：誘電バリア放電)プラズマアクチュエータ（以下 PA）と呼ばれる流体制御デバイス[1, 2]が世界的に注目され，多くの研究がなされている．国内でも 10 年前に日本機械学会にプラズマアクチュエータ研究会が設けられて以来，学術界のみならず，産業界も実用化に向けて研究開発を進めている[3]．

PA は，流れに局所的な流体変動を与えることで周囲の大きな流れを制御するマイクロ流体制御デバイスで，2 枚の電極とそれに挟まれた誘電体からなる単純な構造を持つ（図 1）．電極間に高周波・高電圧の交流電圧を印加することで露出電極周辺にプラズマが発生し，非定常な流体変動が誘起される．薄さは数百マイクロメートルと非常に薄く軽量で，取り付けによる流体機器形状への影響が極めて小さいために既存システムへの導入が容易であることが特徴である．印加電圧の ON/OFF に対して瞬時に応答する点，機械的な駆動部を持たない，消費電力が少ないなど多数の利点を有する．

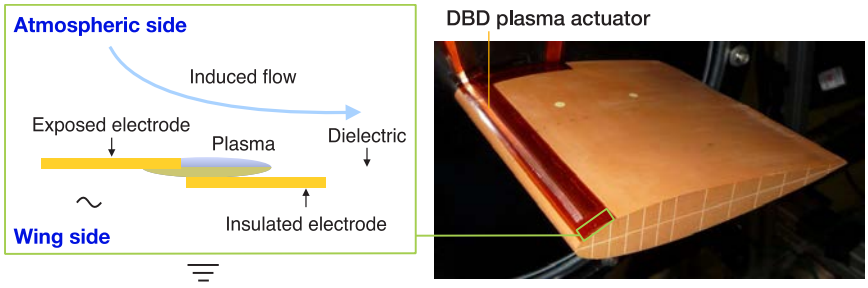
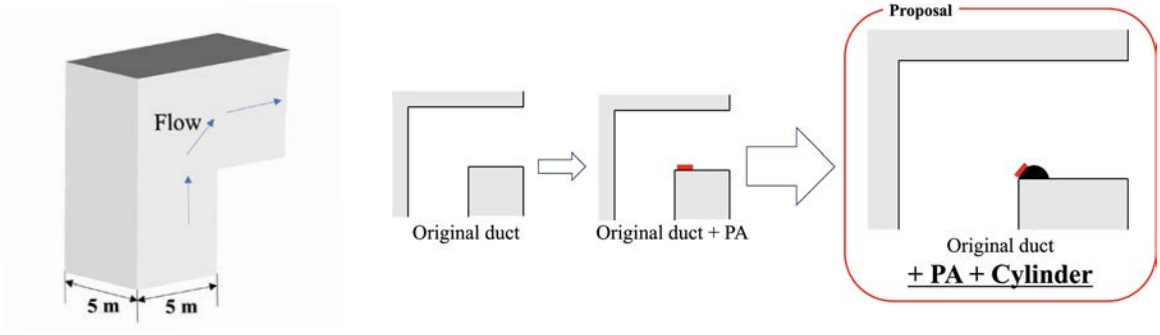


図 1：DBD プラズマアクチュエータの構造と翼型への設置例

PA の研究は航空宇宙工学分野から広まったこともあり，航空機の翼[4-10]，風力発電タービン[11, 12]，ガスタービン[13]といったようないわゆる流線型のような緩やかに形状変化する物体上に生じる剥離を対象とした研究が多い．近年では自動車[15-17]を対象とした研究もなされているが，90 度折れ曲がった物体の角部などで剥離する流れに対して PA を適用した研究は少ない．そこで，我々のグループでは角部で強く剥離するような流れの剥離を PA で制御することに初期的検討として取り組んだが，翼型のように大きく剥離する流れの制御で得られるような明確な効果は得られないことがわかっている．本研究ではこのような物体角部で急激に剥離する流れを制御するために PA に加えて物体角部の剥離を生じる部分に半円柱形状物体を付加することで，できるだけ流れを物体表面に沿うように変え，半円柱物体から緩やかに剥離する流れを更に PA によって制御することを狙う．具体的には図 2a に示すような曲がり管内に生じる剥離を半円柱物体と PA を併用することで制御することを試みる（図 2b）．



(a)　　(b)

図 2: 曲がり管流れへの半円柱物体と PA の適用

2.　問題設定と計算条件

2.1　曲がり管流れの流れ条件，計算条件

解析対象は図 2 のようにL 字に折れ曲がったトンネルの換気に使われるような大型のダクトを対象とし，断面は 5.0 m × 5.0 m の正方形形状とする．内部を流れる流体は空気を想定し，流速 24 m/s で曲がり管下部から直角曲がり部を経て図中（図 2a）右側へと送風される．図 2b のように，内側の角から下流へ 5.0 m の位置で流路は急拡大しており，これは大気開放を想定したものである． 角に設置する半円柱物体は半円柱とし，その直径はダクトの幅の 1/8 とした．半円柱物体は，円柱の端が曲がり管の角と一致するように設置する．代表長さを 0.1 m，代表速さを 24 m/s とし，レイノルズ数は 151,200 である．

表 1 に計算ケースを示す．それぞれの計算ケースに対応する半円柱物体と PA の設置位置の模式図を図 3 に示す．事前検討として，元々の曲がり管形状（図 3a）である Original duct に対して，PA のみを貼り付けた数値計算を行い，効果的な設置位置を検討した（図 3b の PA1 から PA4）．その中で最も効果があった貼り付け位置のものを Original duct+UpstreamPA とする．半円柱を取り付けたケースを Cylinder，半円柱物体と PA を取り付けたケースを表 1 に示す通りに設定する．PA の位置とて Upstream は剥離点より上流，Downstream は下流に設置したことを表している．具体的なそれぞれの位置は図 3d に示す通りである．

表 1: 計算ケース

	半円柱物体の有無	PA の位置	PA の強さ (D_c)
Original duct	No	N/A	N/A
Original duct+UpstreamPA (PA1)	No	Upstream	0.035
Cylinder	Yes	N/A	N/A
Cylinder+UpstreamPA	Yes	Upstream	0.035
Cylinder+UpstreamPA_Strong	Yes	Upstream	0.35
Cylinder+DownstreamPA	Yes	Downstream	0.035
Cylinder+DownstreamPA_Strong	Yes	Downstream	0.35

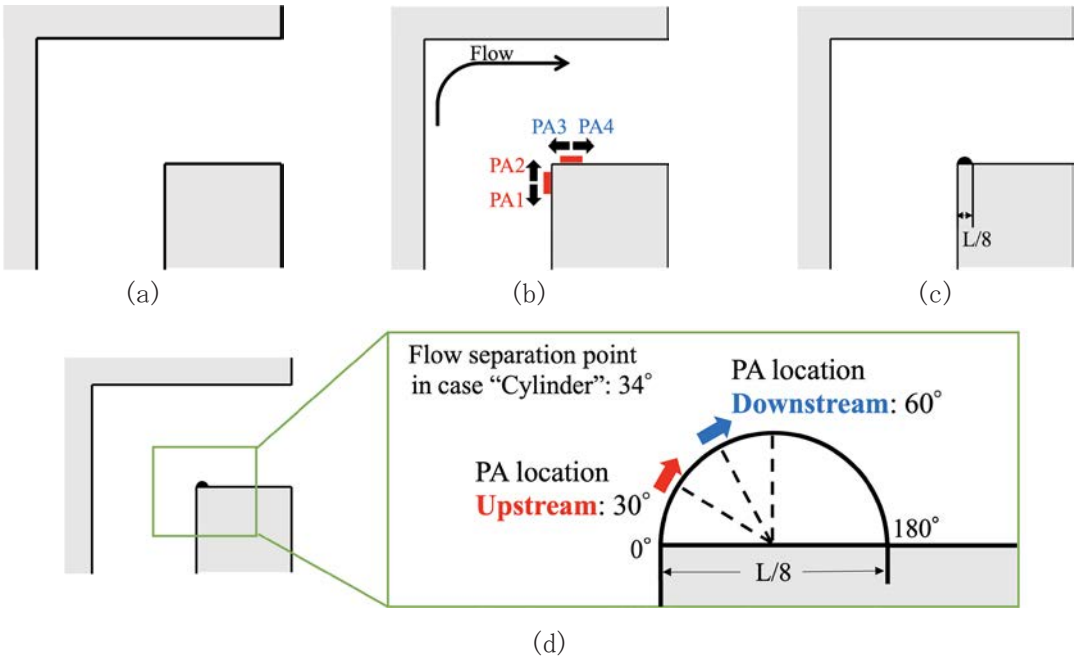


図 3: 計算対象概略図

2.2　プラズマアクチュエータのモデル化

PA の流体計算への導入は支配方程式に PA を模擬した体積力項を導入することで行う．PA の体積力モデルはいくつか提案されているが，ここでは Suzen と Huang ら[18]の定常体積モデル（以下 Suzen モデル）を非定常に拡張した体積モデルを用いる．モデルの数値計算への導入方法の詳細，その信頼性，および数値計算の妥当性については既出論文を参照されたい[19, 20]．PA の駆動は連続駆動とし，周波数は代表速度と代表長さを基準とした無次元周波数で 60 である．アクチュエータの出力を決めるパラメータ *Dc* は表 1 に示したように，0.035 と 0.35 を用いる． *Dc*=0.35 はこれまで行ってきた翼流れ制御[8]において一様流とアクチュエータの誘起流速の比が同程度となる現実的な値であり，いずれも実際の PA で出力可能な範囲の値となっている．

2.3　計算手法

3 次元圧縮性 Navier-Stokes 方程式を支配方程式とし，空間の離散化方法として有限差分法を選択し，数値流束の評価には MUSCL 法[21]により高次精度化した SHUS（Simple High-resolution Upwind Scheme）[22]を用いた．時間積分は 2 次精度 3 点後退差分を 3 回の内部反復を用いた ADI-SGS 陰解法[23]で解いた．内部反復の導入によって大きな時間ステップ幅でも時間精度が維持されるようにしている．代表速度と代表長さを用いた無次元時間刻みは 2.5×10^{-4} とした．

2.4　計算格子と境界条件

図 4 に計算格子を示す．計算格子は曲がり管流れを解く Zone 1（図 4b，黒色）の格子と半円柱物体周りの流れを解く Zone 2（図 4c，4d，青色），PA 近傍の流れを解く Zone 3（図 4c，4d，赤色）の格子の計 3 つのゾーンからなり，総格子点数は約 500 万点である．曲がり管出口から外部境界までの距離は 12.5L とした．奥行き方向には周期境界条件を用い，それ以外の壁面は滑りなし壁条件を適用した．流入境界では圧力を外挿し，他の物理量を流入速度の条件で固定した．

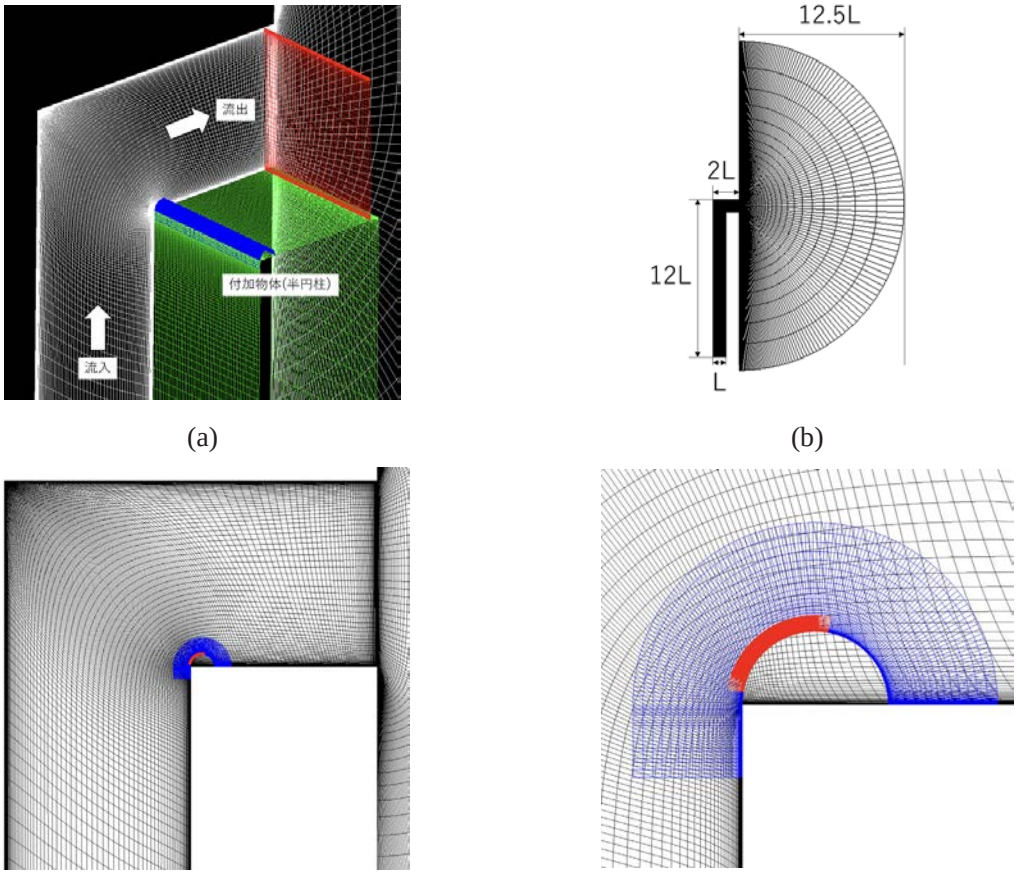


図 4: 計算格子

2.5　計算機に関して

計算には，東北大サイバーサイエンスセンターのスーパーコンピュータ AOBA のサブシステム AOBA-A を利用した．16VE を利用し，分割した格子を各 VE に割当てて計算を実施した．各 VE 間の通信には MPI を用い，VE 内は自動並列を行うハイブリッド並列を用いた．1 ケース(200 万ステップ)の計算時間は約 22 時間である．

3.　計算結果

3.1　瞬間流れ場

図 5 に 500 万ステップ時点の瞬間速度場のうち代表的なものを示す．面塗りは速度の絶対値 $||u||$ を代表速度 $||u_\infty||$ で無次元化した値を示す．各図では左下から空気が流入し，直角な角部を通って流れが 90 度折れ曲がった後に各図の右中央にある大気開放部へと流出する．いずれの図においても曲がり部直後から大気開放部までの間に青色の領域が存在しており，流れの剥離が生じているのがわかる．Original duct では角部において流れが特に強く剥離し，流路が狭まるために大気開放部付近で速度の増加がみられる．一方で Cylinder では，Original duct に比べて角に設置した半円柱物体表面に流れが付着し下流の剥離領域がやや小さくなっていることがわかる．

本研究では半円柱物体は PA の効果を補助するために取り付けたが，半円柱物体のみを取り付けた場合でも大きな剥離制御効果が得られた．Cylinder と Cylinder+UpstreamPA の瞬間場の比較では明確な違いは見受けられない．

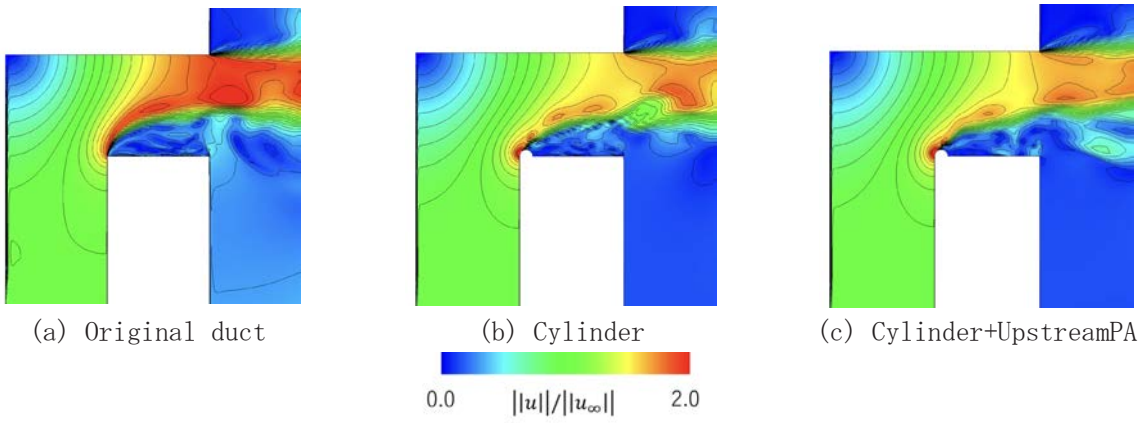


図 5:瞬間速度場

3.2　平均流れ場

400 万ステップから 600 万ステップの間の流れを時間平均し，さらに紙面奥行方向（周期境界条件を課した方向）に空間平均を行った平均速度流れを図 6 に示す．面塗りは速度の絶対値，黒い線は流線を表している．Original duct（図 6a）では角から流れは剥離し，大気開放部付近で流速増加が生じている．半円柱物体を設置した Cylinder（図 6b）では，角に設置した半円柱物体表面に流れが付着することで剥離が抑えられており，流速の増加も抑制されている．一方で，半円柱物体による変化と比べると PA を用いた場合の流れの変化は図 6 からはっきりとはわからない．そこで，半円柱物体付近の流れ場を拡大した流れ場を図 7 に示す．

図 7 は図 6 と同様に速度の絶対値の平均場と流線を示している．図 7 に描かれた矢印の位置は PA のおおよその設置位置を，矢印の太さは PA の強さの大小を示す．各ケースの間で半円柱物体からの剥離剪断層の角度に違いがあることが流線から見てとれる．Cylinder では剥離剪断層がちょうど図右上の角付近へと伸びている．PA を用いたケースのうち Cylinder+UpstreamPA，Cylinder+ DonwstreamPA，Cylinder+DonwstreamPA_Strong では剥離剪断層はやや水平に近づいており，PA によって流れが付着し，剥離が抑えられているのがわかる．反対に，Cylinder+UpstreamPA_Strong では剥離剪断層が垂直に近づいており，PA によって流れの剥離が促進されているのがわかる．

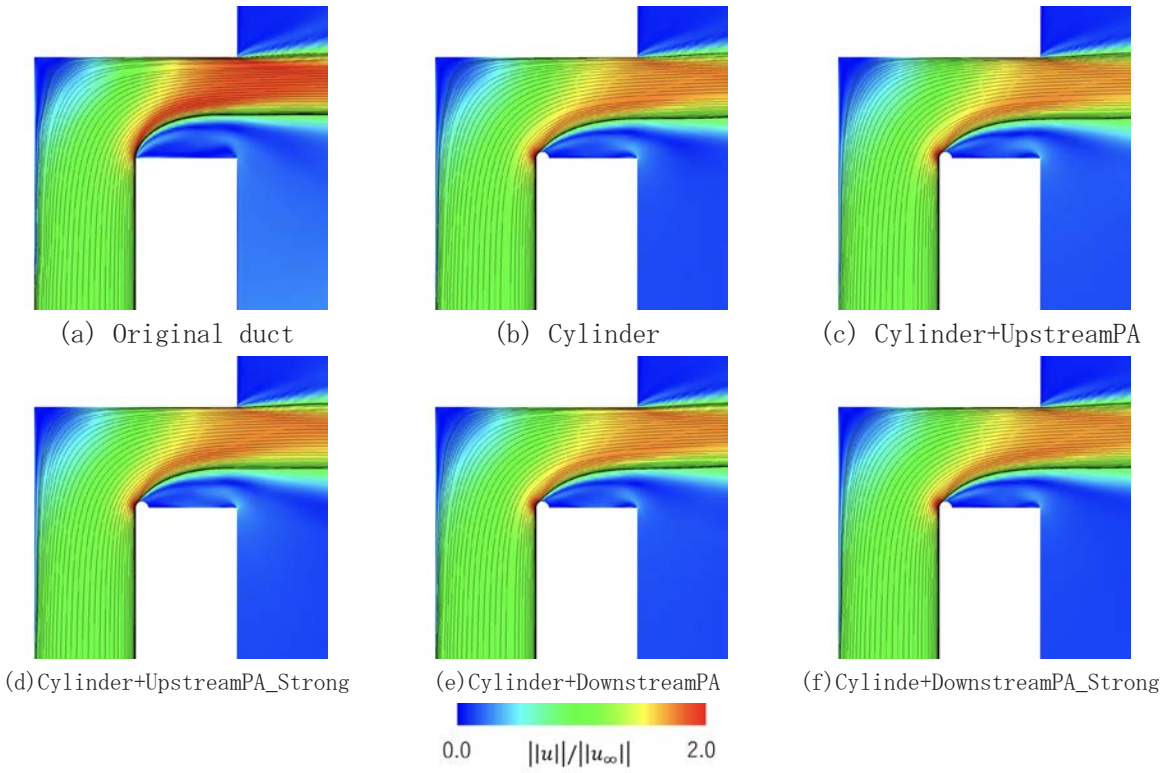


図 6: 平均速度場と流線

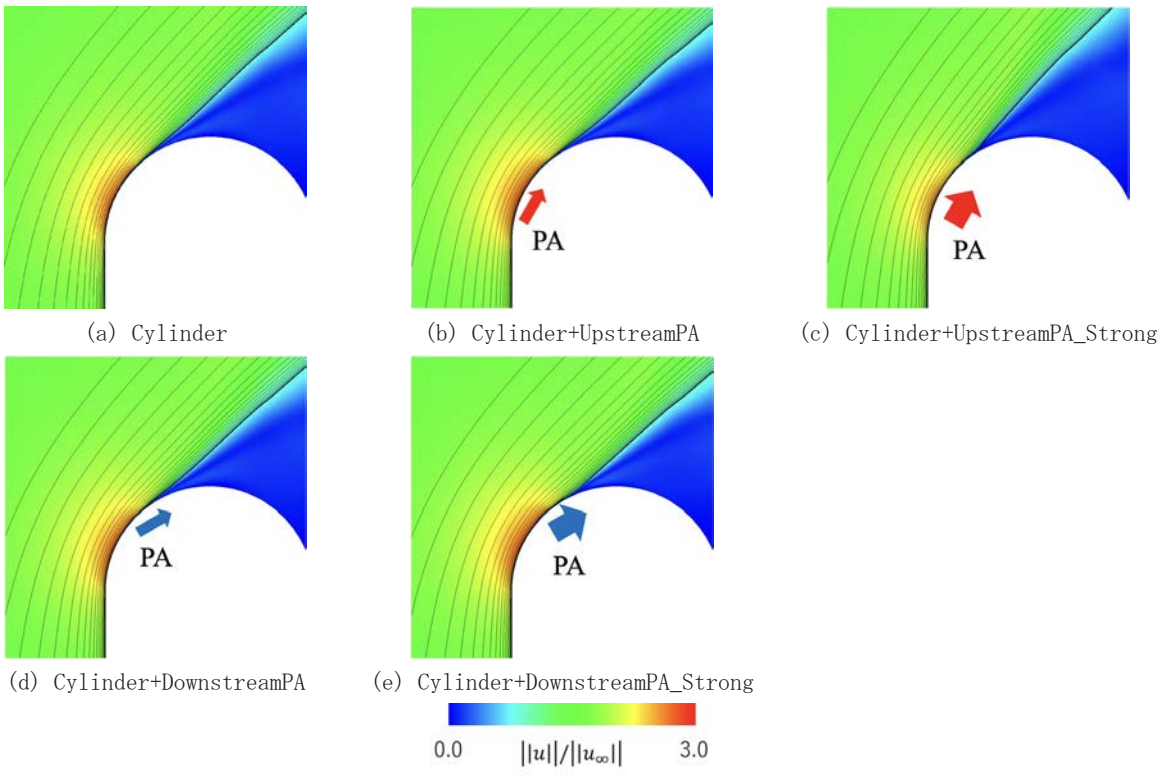


図 7: 半円柱物体表面付近の時間平均速度場と流線

各ケースの流れの剥離を定量的に評価するために半円柱物体表面付近の運動量厚さで比較する．図 8 の縦軸は半円柱物体表面の運動量厚さであり，横軸は半円柱物体表面における x 座標をダクトの幅で無次元化した座標値である．灰色の点線はそれぞれおおよその PA の設置位置を示している．縦軸の運動量厚さの増加は剪断層が物体から離れることに対応する．運動量厚さの小ささから剥離の抑制度で順位付けをすると以下のようになる．

1. Cylinder+DownstreamPA_Strong
2. Cylinder+UpstreamPA
3. Cylinder+DownstreamPA
4. Cylinder
5. Cylinder+UpstreamPA_Strong

図 7 で示した様に，Cylinder+UpstreamPA_Strong では剥離が促進され，他の PA を設置したケースでは剥離が抑制されていることがわかる．PA による誘起流が比較的弱いケースである Cylinder+UpstreamPA と Cylinder+DownstreamPA の結果を比較すると，Cylinder+UpstreamPA の方が運動量厚さは小さく，剥離が抑制できている．UpstreamPA と DownstreamPA は PA の設置位置が剥離点位置の上流か下流という違いに加えて剥離点位置と PA の体積力が最大になる位置に違いがある．UpstreamPA は剥離点位置より 4 度ほど上流の位置であるが，DownstreamPA の場合は剥離点位置より 26 度ほど下流の位置である．従って，PA による誘起流が主流と比べて弱い場合は，剥離点と近い位置に PA を設置することが効果的であったと考えられる．一方で，PA による誘起流が強い場合の Cylinder+UpstreamPA_Strong と Cylinder+DownstreamPA_Strong の結果を比較すると，Cylinder+UpstreamPA_Strong は Cylinder+DownstreamPA_Strong よりも運動量厚さは大きく，更に PA を設置していない Cylinder よりも剥離は大きくなっている．その理由はいくつが考えられるが，ひとつは Cylinder+UpstreamPA_Strong では半円柱表面に設置された PA から接線方向に強く流れが誘起されるため，慣性力が強すぎて剥離した可能性，もう一つは PA が誘起する壁面方向への誘起速度が強く，流れが壁に衝突することで圧力が上昇し，逆圧力勾配を生じて流れが剥離した可能性などである．このように，PA による誘起流が強力な場合，元の流れの剥離点位置に加えて，PA による誘起流の方向についても考慮して設置位置を決める必要がある可能性がある．

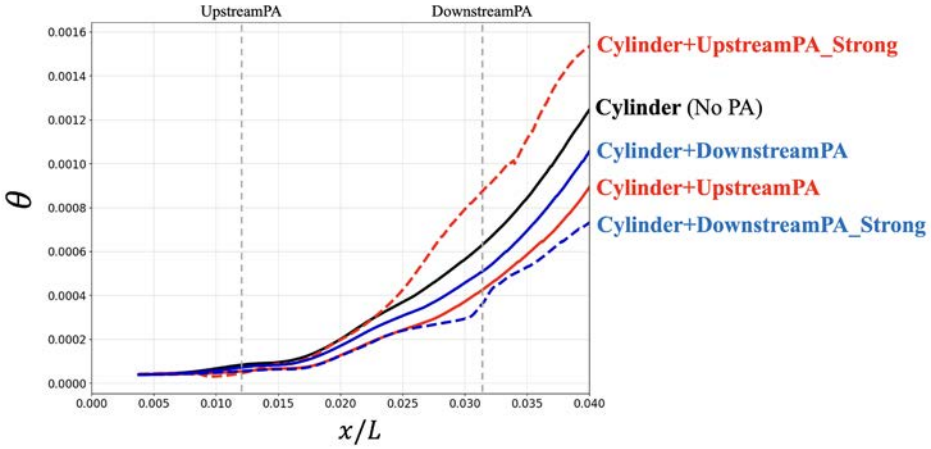


図 8: 半円柱物体表面における運動量厚さ

半円柱物体による PA の制御効果の増減を曲がり部前後の総圧損失で評価する．図 9 に各ケースの総圧損失を示す．総圧損失は流入部の平均総圧に対する流出部の平均総圧の差で求めた．曲が

り部内側の角から流路幅の長さだけ上流の位置を流入部の断面として用い，同様に角から流路幅だけ下流の位置を流出部の断面として用いた．また，流れ場の時間平均を行った後に奥行き方向にも平均処理を行った．Original duct と Cylinder を比較すると，半円柱物体の付加で総圧損失が 28.4%減少したことがわかる．また，Cylinder と半円柱物体に PA を設置した 4 ケースを比較すると，Cylinder+UpstreamPA_Strong を除いて PA の設置によって更に総圧損失が減少したことがわかる．Cylinder+DownstreamPA_Strong においては最も総圧損失が減少し，Cylinder に比べて 3.5%の減少である．一方で，Cylinder+UpstreamPA_Strong では Cylinder に比べて総圧損失が 3.5%増加した．総圧損失の小ささで順位付けをすると以下のようになる．

1. Cylinder+DownstreamPA_Strong
2. Cylinder+UpstreamPA
3. Cylinder+DownstreamPA
4. Cylinder
5. Cylinder+UpstreamPA_Strong

この順位は運動量厚さから評価した剥離抑制効果の順位と一致しており，本研究の条件下では，曲がり部前後の総圧損失と半円柱物体表面における剥離の様子との間に相関関係が見られることがわかる．

最後に，総圧損失の減少度を用いて PA 単体を流路に設置した場合と半円柱物体と PA を併せて設置した場合を比較する．元の曲がり管形状に PA 単体を設置した場合の最良ケースは Original duct+UpstreamPA であり，総圧損失の減少は 0.1%である．半円柱物体と PA の両方を流路に設置した場合，最良ケースは Cylinder+DownstreamPA_Strong であり，半円柱物体のみの取り付けに比べて総圧損失の減少は 3.5%である．PA 出力の弱い Cylinder+UpstreamPA であっても，総圧損失は 2.8%減少する．元の直角な流路に PA のみを設置した場合の総圧損失の減少が 0.1%であることに比べると，半円柱物体と PA を併用した場合に PA の効果はその誘起流の強さに関わらず増強されていると言える．従って，PA の設置条件や駆動条件にはよるが，PA による制御効果を得にくい流路に対し，半円柱物体と PA を併用することで PA の効果を増幅できる可能性があることが本研究でわかった．

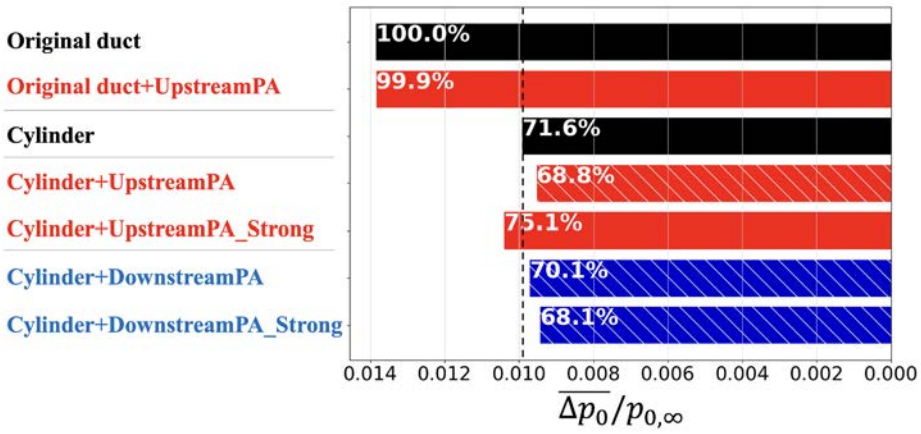


図 9：総圧損失変化の変化

3.4 今後の研究計画

本研究では 3 次元計算を行ったものの，比較的粗い計算格子を用いた低次精度の数値計算であるために，乱流のような複雑な流体現象は捉えられていない．より詳細に流れ場の議論を行うために，今後高忠実なシミュレーションを実施したいと考えている．また，今回は曲がり管内の流れ，すなわち内部流を対象としたが，外部流に対しても半円柱と PA の組み合わせで流れ制御が効果的に行えるかどうかの検証を現在行っている．

4. まとめ

本研究では曲がり管内の流れに対して，剥離を生じる角部付近に半円柱物体を設置し，PA と併用することで流れの剥離を抑制することを考え，数値計算を用いて剥離制御効果の評価を行った．シミュレーションの結果，半円柱物体を設置しただけでも角部の剥離を抑制することができ，PA を併用することでさらに剥離を抑えることがわかった．総圧損失による評価では半円柱の設置によって総圧損失は 28.4%減少し，さらに半円柱に PA を駆動することで半円柱物体のみを設置した場合に比べて最大 3.5%現象することがわかった．PA の設置位置に関する検討も行った結果，PA の出力が比較的弱い場合は剥離に対して上流側に PA を取り付けた場合が効果的であったが，PA の出力が強い場合は剥離点より下流に PA を設置した方が剥離を抑えられることがわかった．ただし，今回の計算では，PA を上流に設置したケースの方が PA の体積力が最大となる位置が剥離点に近いことに留意したい．また，壁面方向に向かう方向の流れの誘起についても注意が必要である．PA の強さと設置位置が剥離制御効果に与える影響に関しては今後，高忠実なシミュレーションの実施によってより詳細な議論を行いたいと考えている．

謝辞

ここに記載させて頂いた成果は、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ものである．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頂いた．センターの皆様にこの場を借りて謝意を示したい．

参考文献

[1] T. C. Corke, C. L. Enloe, and S. P. Wilkinson, “Dielectric Barrier Discharge Plasma Actuators for Flow Control,” Annual Review of Fluid Mechanics, Vol. 42, pp. 505–529, 2010.
[2] T. C. Corke, M. L. Post, and D. M. Orlov, “Single Dielectric Barrier Discharge Plasma Enhanced Aerodynamics: Physics, Modeling and Applications,” Experiments in Fluids, Vol. 46, No. 1, pp. 1–26, 2009.
[3] 日本機械学会，プラズマアクチュエータ研究会，http://plasma-actuators.jp/
[4] M. L. Post and T. C. Corke, “Separation Control on High Angle of Attack Airfoil Using Plasma Actuators,” AIAA Journal, Vol. 42 No.11,2004.
[5] N. Benard, J. Jolibois, and E. Moreau, “Lift and Drag Performances of an Axisymmetric Airfoil Controlled by Plasma Actuator,” Journal of Electrostatics, Vol. 67, No. 2-3, pp. 133–139, 2009.
[6] K. Asada, Y. Ninomiya, A. Oyama, and K. Fujii, “Airfoil Flow Experiment on the Duty Cycle of DBD Plasma Actuator,” 47th AIAA Aerospace Sciences Meeting, The New Horizons Forum and Aerospace Exposition, Orlando, Florida, January 2009.
[7] K. Fujii “High-Performance Computing Based Exploration of Flow Control with Micro Devices” Philosophical Transaction A, The Royal Society, Vol. 372, Article ID 20130326, 2014.

[8] M. Sato, H. Aono, A. Yakeno, T. Nonomura, K. Fujii, K. Okada, and K. Asada, “Multifactorial Effects of Operating Conditions of Dielectric Barrier Discharge Plasma Actuator on Laminar Separated Flow Control,” AIAA Journal, Vol. 53, No. 9, 2015.
[9] H. Aono, S. Kawai, T. Nonomura, M. Sato, K. Fujii and K. Okada, “Plasma-Actuator Burst-Mode Frequency Effects on Leading-Edge Flow-Separation Control at Reynolds Number 2.6×10^5,” AIAA Journal Vol. 55, pp. 3789-3806, 2017.
[10] K. Fujii, “Three Flow Features behind the Flow Control Authority of DBD Plasma Actuator: Result of High-Fidelity Simulations and the Related Experiments,” Applied Science 2018, Vol. 8, Issue 4, 2018.
[11] H. Matsuda, M. Tanaka, S. Goshima, K. Amemori, M. Nomura and T. Osako, “Experimental Study on Plasma Aerodynamic Control for Improving Wind Turbine Performance,” Asian Congress on Gas Turbins 2012, Shanghai, P. R. China, August 2012.
[12] D. Greenblatt, A. B. Harav, and H. M. Vahl, “Dynamic Stall Control on a Vertical Axis Wind Turbine Using Plasma Actuators,” AIAA Journal, Vol. 52, No. 2, pp. 456–461, 2014.
[13] D. P. Rizzetta and M. R. Visbal, “Simulation of Plasma-based Flow Control Strategies for Transitional Highly Loaded Low-Pressure Turbines,” 37th AIAA Fluid Dynamics Conference and Exhibit, Fluid Dynamics and Co-located Conferences, Miami, Florida, June 2007.
[14] K. Shimizu, T. Nakajima, S. Sekimoto, K. Fujii, T. Hiraoka, Y. Nakamura, T. Nouzawa, J. Ikeda and M. Tsubokura, “Aerodynamic drag reduction of a simplified vehicle model by promoting flow separation using plasma actuator, JSME Mechanical Engineering Letters, Bulletin of the JSME, Vol.5, No. 19-00354,2019.
[15] Z. Hui, X. Hu, P. Guo, Z. Wang and J. Wang, “Separation Flow Control of a Generic Ground Vehicle Using an SDBD Plasma Actuator,” MDPI, Open Access Journal, vol. 12, issue 20, pp. 1-14, 2019.
[16] S. Shadmani, S. M. Mousavi Nainiyan, M. Mirzaei, R. Ghasemiasl and S. G. Pouryoussefi, “Experimental Investigation of Flow Control over an Ahmed Body using DBD Plasma Actuator,” Journal of Applied Fluid Mechanics, Vol. 11, No. 5, pp. 1267-1276, 2018.
[17]浅田健吾, 藤井孝藏, “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,” SENAC, Vol. 53, No. 1, Jan. 2020.
[18] Y. B. Suzen and P. G. Huang, “Simulations of Flow Separation Control using Plasma Actuators,” 44th AIAA Aerospace Sciences Meeting and Exhibit, Aerospace Sciences Meetings, Reno, Nevada, January 2006.
[19] K. Asada, T. Nonomura, H. Aono, M. Sato, K. Okada, K. Fujii, “LES of Transient Flows Controlled by DBD Plasma Actuator over a Stalled Airfoil,” International Journal of Computational Fluid Dynamics, Vol. 29, 2015.
[20] H. Aono, S. Sekimoto, M. Sato, A. Yakeno, T. Nonomura, and K. Fujii, “Computational and Experimental Analysis of Flow Structures Induced by a Plasma Actuator with Burst Modulations in Quiescent Air” Bulletin of the JSME Mechanical Engineering Journal Vol. 2, No. 4, 2015.
[21] Bram van Leer. Towards the ultimate conservation difference scheme. iv. a new approach to numerical convection. Journal of Computational Physics, Vol. 23, No. 3, pp. 276–299, 1977.
[22] E. Shima and T. Jounouchi. Role of cfd in aeronautical engineering (no.14) -ausm type upwind schemes-. In Proceedings of the 14th NAL Symposium on Aircraft Computational Aerodynamics, pp. 7–12. NAL, 1997.
[23] K. Fujii, “Simple Ideas for the Accuracy and Efficiency Improvement of the Compressible Flow Simulation Methods.” Paper presented at International CFD Workshop for Super-sonic Transport Design, Tokyo, March, 1998.

［共同研究成果］

格子ガス法流体解析における背景粒子場モデルの活用と将来性

――――“仮想光子場”の中でブラウン運動する仮想粒子への達人操作

松岡　浩
技術士事務所 AI コンピューティングラボ

筆者は、東北大学サイバーサイエンスセンターの共同研究公募制度により、令和 3 年度から 5 年度までの 3 年間で「リカレント型ビット演算による縦渦挙動のマルチスケール創発解析」を実施してきた。縦渦は、高レイノルズ数状態の流れで発生するため、その数値シミュレーションに用いる流体は、非常に小さい正の値の粘性を発現する必要がある。筆者は、これまで 10 年以上にわたり、格子ガス法流体解析において、「望むだけ小さい粘性を、極めて簡単な計算手順で発現できる粘性制御モデル」を夢みて追求してきた。本稿では、その最新版である“背景粒子場モデル”と名付けたモデルに基づく粘性制御手法の検討状況を紹介する。また、円柱後流における縦渦創発のシミュレーションなどへの適用事例を定性的に考察し、その手法の将来性について言及する。

１．２つの“背景粒子場モデル”（“仮想反粒子場モデル”と“仮想光子場モデル”）

（１）仮想反粒子場モデル

“背景粒子場モデル”として、２つのものを考える。ひとつは、“仮想反粒子場モデル”であり、“仮想反粒子”が、流体の存在する空間中の全ての格子点にある程度存在していると考える。流体自体を模擬している通常の仮想粒子が、これらと相互作用を起こすことによって低粘性流体の挙動を発現させる。当該モデルにおける“仮想反粒子”は、負の質量（$m<0$）をもつ仮想粒子であり、従って、この粒子がある向きに正の速さ（$v>0$）をもっていれば、当該粒子は、逆向きの運動量（$mv<0$）と負のエネルギー（$\frac{1}{2}mv^2<0$）をもつことになる。正の質量をもつ通常の仮想粒子が“仮想反粒子”と相互作用を起こせば、両粒子の“ペア消滅”が生じる。このとき、質量も運動量もエネルギーも正負が打ち消しあうので真空状態が残る。逆に、真空状態から、仮想粒子と“仮想反粒子”のペアを、必要に応じて生成させることができる。（cf. **図１**）

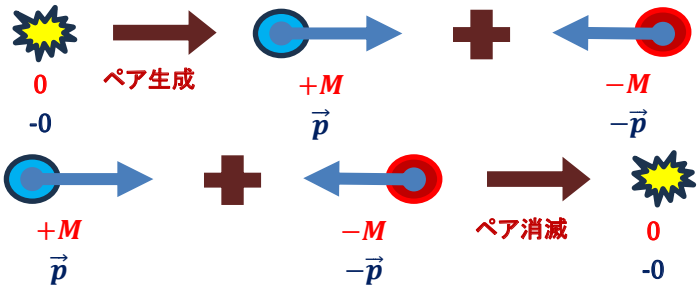


図１．仮想粒子と“仮想反粒子”のペア生成消滅過程

当面のシミュレーションニーズにおいては、“仮想反粒子”は、各格子点に拘束させる。格子点間を移動できるのは、通常の仮想粒子のみだと考える。そして、流体の質量や運動量やエネルギーを集計して疎視化を行う場合、仮想粒子のみを合計してマクロな物理量を導出する。“仮想反粒子”は集計に含めず、目に見えない背景空間であるかのように扱う。従って、仮想粒子と“仮想反粒子”のペアの生成消滅過程を導入することで、必要に応じて、仮想粒子を目の前に生成させたり、目の前から消滅させたりすることができる。これは、シミュレーション上出会ういろいろな問題を解決するのにかなり役立つ。ただし、ひとつだけ重要な制限がある。「仮想粒子と“仮想反粒子”の“ペア生成”と“ペア消滅”は、疎視化を行う時空間のスケールにおいて平衡して生じ

させる」ことが必要である。これによって、仮想粒子の挙動だけを集計して疎視化した場合でも、質量と運動量とエネルギーの保存則を破らないことが保証できる。

なお、“仮想反粒子”というここでの呼称は、物理学における“反粒子”の概念からの連想であるが、物理学における“反粒子”は正の質量をもつなど全てが同様ではないことに注意されたい。

【“仮想反粒子場モデル”の発想】

筆者が“仮想反粒子”の概念を目にしたのは、Robert P. Bosch, Jr による MIT の論文「A Multigrid Algorithm for Lattice Gases（格子ガスのためのマルチグリッドアルゴリズム)」においてである。流体解析シミュレーションの計算機負担を軽くするため、格子ガス法においても、流れの中の物体近傍では格子点間隔を小さくし、物体から遠く離れた位置では格子点間隔を大きくしたい。このとき、格子点間隔が異なる複数の格子を使用することになる。細かい格子と粗い格子の接続領域では、質量、運動量、エネルギーの保存をうまく維持しながら、「細かい格子上を動く複数の“軽い仮想粒子”」と「粗い格子上を動くひとつの“重い仮想粒子”」を等価交換しなければならない。例えば、“重い仮想粒子”の質量が“軽い仮想粒子の”８倍であった場合、ある向きの速度をもつ重い粒子１個が粗い格子から接続領域にやってきたとすれば、接続領域では“重い仮想粒子”を１個消滅させ、同じ向きの速度をもつ“軽い仮想粒子”を８個生成させて細かい格子に送り出せばよい。しかし、その逆には工夫が必要である。例えば、細かい格子から同じ向きの速度をもつ“軽い仮想粒子”６個が接続領域にやってきたときは、とりあえず、同じ向きの速度をもつ“重い仮想粒子”１個を粗い格子に送り出し、接続領域には、２個の同じ向きの速度をもつ“軽い仮想反粒子”を残しておく。これらの“軽い仮想反粒子”は、その後、同じ向きの速度をもつ“軽い仮想粒子”がくれば、その個数分、ペア消滅させる。こうして、疎視化を行う時空間スケールでは、質量、運動量、エネルギーの保存を成立させることができる。本稿のモデルは、この“仮想反粒子”の存在する領域を、接続領域から流体が存在する全領域に広げたものである。

（２）仮想光子場モデル

もうひとつは、“仮想光子場モデル”である。“仮想光子”が、流体の存在する空間中の全ての格子点にある程度存在していると考える。流体自体を模擬している通常の仮想粒子が、これらと相互作用を起こすことによって低粘性流体の挙動を発現させる。ただし、当面のシミュレーションニーズにおいては、“仮想光子”は仮想粒子のうち“静止している仮想粒子”とだけ相互作用を起こすと仮定する。“仮想光子”は、まさに通常の光のように考え、目に見えない背景空間をなす。静止した仮想粒子を動かしたい時は、背景空間から“仮想光子”を拾い上げ、それと合体して運動量をもらう。運動している仮想粒子を静止させたい時は、背景空間に“仮想光子”を分離して、運動量を返却する。“仮想光子”は、仮想粒子がもつような質量はもたないが、運動量とエネルギーをもつ。そして、流体が存在する領域においては、“仮想反粒子モデル”の場合と同様に、「静止仮想粒子と“仮想光子”の合体と分離は、疎視化を行う時空間のスケールにおいて平衡して生じさせる」ことが必要である。（cf. **図２**）

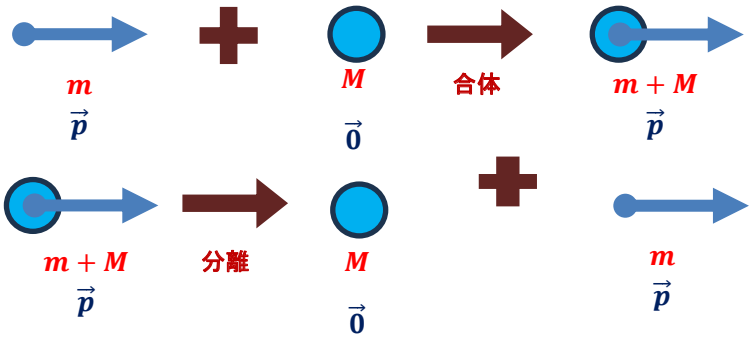


図２．“静止仮想粒子”と“仮想光子”の合体分離過程
（注：上図で、*m*は“仮想光子”がもつエネルギーに対応する質量）

【“仮想光子場モデル”の発想】

“静止仮想粒子”が、全空間中に満ちている“仮想光子”と合体や分離を引き起こす過程のイメージは、花粉が多数の水分子の中で“ブラウン運動”するイメージを連想させる。疎視化を行う時空間スケールにおいて、合体と分離が平衡して生じていれば、花粉（＝“静止仮想粒子”）は、静止状態を保持してみえるはずである。しかしながら、ときどき同じ向きに連続して水分子（＝“仮想光子”）が花粉（＝“静止仮想粒子”）に衝突（＝相互作用）すれば、花粉（＝“静止仮想粒子”）は揺動する。こう考えると、流体を模擬している仮想粒子の揺動運動についても、非平衡統計力学における揺動散逸定理（グリーン-久保公式）[1]のようなものが成立し、粘性係数に直接関連づけられると思われる。しかし、ここでは、この詳細に深入りする必要はない。「グリーン-久保公式は流体の粘性係数を与える式であり、その導出の前提には“揺動力”の仮定があって、その大きさが粘性係数の値を左右する」という点だけに注目する。

流体場には、いつも系を乱そうとする“揺動力”が働いている。“揺動力”はあくまで揺らぎなので平均したらゼロでなければならない。あるときにプラスの揺動力が作用したとすれば、平均的には、次にはマイナスの揺動力が作用するので、その効果はゼロになる。しかし、たまたまほぼ同じ時刻に同じ場所で同じ向きの揺動力が続けて２回以上作用する場合もあり、このとき、“揺動力”は効果をもつ[1] 。“仮想光子場モデル”では、“プラスの揺動力”が「“仮想光子”が“静止仮想粒子”に衝突合体して“運動仮想粒子”を生成する過程」に対応し、“マイナスの揺動力”が「“運動仮想粒子”が“仮想光子”を分離して“静止仮想粒子”を生成する過程」に対応づけられる。従って、“静止仮想粒子”と“同じ向きの運動量をもつ仮想光子”の合体か分離のいずれか一方の過程が続けて２回以上作用する確率が、流体粘性に直接効果を及ぼすはずである。“仮想光子場”に基づく粘性制御手法は、この発想に基づくものであり、非平衡統計力学における流体モデル[1]と整合している。

２．“仮想反粒子場モデル”に基づく粘性制御手法による縦渦挙動創発解析

まず、“仮想反粒子場モデル”に基づく粘性制御を利用した縦渦挙動創発解析の事例について述べる。計算手順の大枠は、通常の格子ガス法の計算手順（“並進移動”及び“衝突散乱”）に加えて、粘性制御を行うための「同期連行過程」と呼ぶ過程を追加したものである。（cf. **図３**）

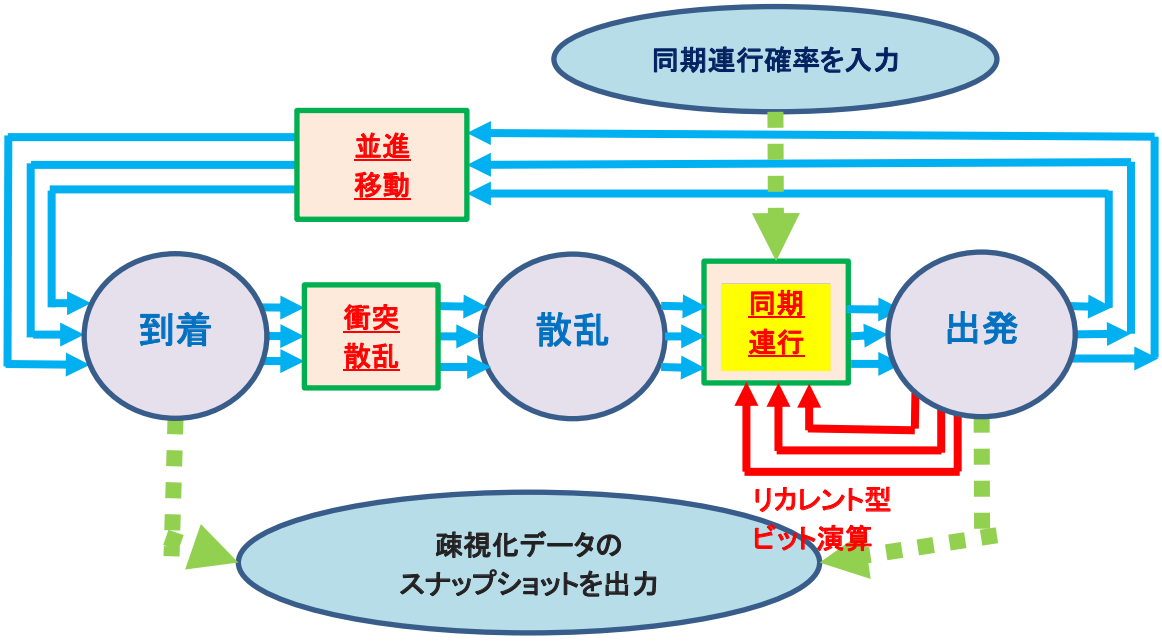


図３．粘性制御のための同期連行過程を組み込んだ計算手順

「同期連行過程」における計算手順は、極めて簡単である。流体領域にあるすべての格子点のすべての“方向”に対して、以下に述べる操作を行えばよい。ただし、ここで“方向”とは、ある向きＤとその逆向きをセットで考えた概念であるとする。

【同期連行過程における計算手順（“仮想反粒子場モデル”の場合)】

１．時刻ｔ－1にＤ向きまたは逆Ｄ向きの速度をもって格子点から出発した仮想粒子、及び、時刻ｔにＤ向きまたは逆Ｄ向きの速度をもって格子点に到着した仮想粒子の、合計４つの仮想粒子だけに注目して、その存否（存在するか否か？）を調べる。

２．４つの仮想粒子うち、時刻ｔ－1 に向きＤの速度をもって格子点から出発した仮想粒子だけが存在すれば、

① 時刻ｔにも向きＤの速度をもつ仮想粒子を出発させるよう努める。
⇒このため、時刻ｔにおける衝突散乱の結果、向きＤに出発する仮想粒子が既に準備されていれば何もしない。準備されていない場合は、向きＤの速度をもつ“仮想反粒子”が存在しない場合に限り、真空からペア生成によって向きＤに出発できる仮想粒子を準備する。

② 時刻ｔに逆Ｄ向きに仮想粒子を出発させないよう努める。
⇒このため、時刻ｔにおける衝突散乱の結果、逆Ｄ向きに出発する仮想粒子が準備されていなければ何もしない。準備されている場合は、逆Ｄ向きの速度をもつ“仮想反粒子”が存在する場合に限り、ペア消滅によって逆Ｄ向きに出発できる仮想粒子を削除する。

３．２の操作は、逆Ｄ向きに対しても同じに実行する。

４．１から３の操作は、確率的に行うとともに、例えば「必ず交互に行う」などの方法により、平衡が成立するように実行する。

以上の計算手順は、詳細な説明は省略するが、下記のようなビット演算で実行できる。

プログラム：

```
AF = AV[D];  SF = ST[D];  RF = RX[D];  MF = MI[D];
AB = AV[D̄];  SB = ST[D̄]; RB = RX[D̄];  MB = MI[D̄];
mkF = (~AF & ~AB &  SF & ~SB & HASA & ~RF & ~MF &  EXC)
     |(~AF & ~AB & ~SF &  SB & HASA &  RF &  MF & ~EXC);
mkB = (~AF & ~AB & ~SF &  SB & HASA & ~RB & ~MB &  EXC)
     |(~AF & ~AB &  SF & ~SB & HASA &  RB &  MB & ~EXC);
EXC   = EXC ^ (mkF ^ mkB);
RX[D] = RF ^ mkF;   MI[D]  = MF ^ mkF;
RX[D̄] = RB ^ mkB;   MI[D̄] = MB ^ mkB;。
```

ここで、

AV[D]：　時刻ｔにＤ向きの速度をもって到着した仮想粒子の存否配列
ST[D]：　時刻ｔ-1にＤ向きの速度をもって出発した仮想粒子の存否配列
RX[D]：　時刻ｔの衝突散乱によって生じたＤ向きの速度をもつ緩和粒子の存否配列
MI[D]：　時刻ｔにＤ向きの速度をもって当該格子点に存在する仮想反粒子の存否配列
AV[D̄]：　時刻ｔに逆Ｄ向きの速度をもって到着した仮想粒子の存否配列
ST[D̄]：　時刻ｔ-1に逆Ｄ向きの速度をもって出発した仮想粒子の存否配列
RX[D̄]：　時刻ｔの衝突散乱によって生じた逆Ｄ向きの速度をもつ緩和粒子の存否配列
MI[D̄]：　時刻ｔにＤ向きの速度をもって当該格子点に存在する仮想反粒子の存否配列
HASA：　同期連行を試みる確率に応じて、HASAの各ビットの「１」の出現数が決まる。
EXC：　平衡が成立するように、交互に処理を行うための工夫。

以上の計算手順により、円柱後流に生じる縦渦の創発挙動を計算した。
本稿に示した計算に限らず、筆者がこれまでに行ってきたシミュレーション研究[2]を通じて、格子ガス法による仮想粒子の衝突散乱モデルを用いた場合でも、円柱後流における縦渦の創発については、「流体粘性を小さく制御し高レイノルズ数状態を維持した場合、円柱背後における剥離泡の崩壊位置が円柱軸方向に激しく振動し、崩壊した剥離泡から発生する流体の円柱軸方向の強い流れが縦渦創発の起源となり、この流れが円柱背面から流下向きに離れて、円柱後流のローラ渦列の間で縦渦として強調されていく様子」を観察できている。 **図４**の右下図は、円柱の下流位置において「発達した縦渦」が現れていることを示している。

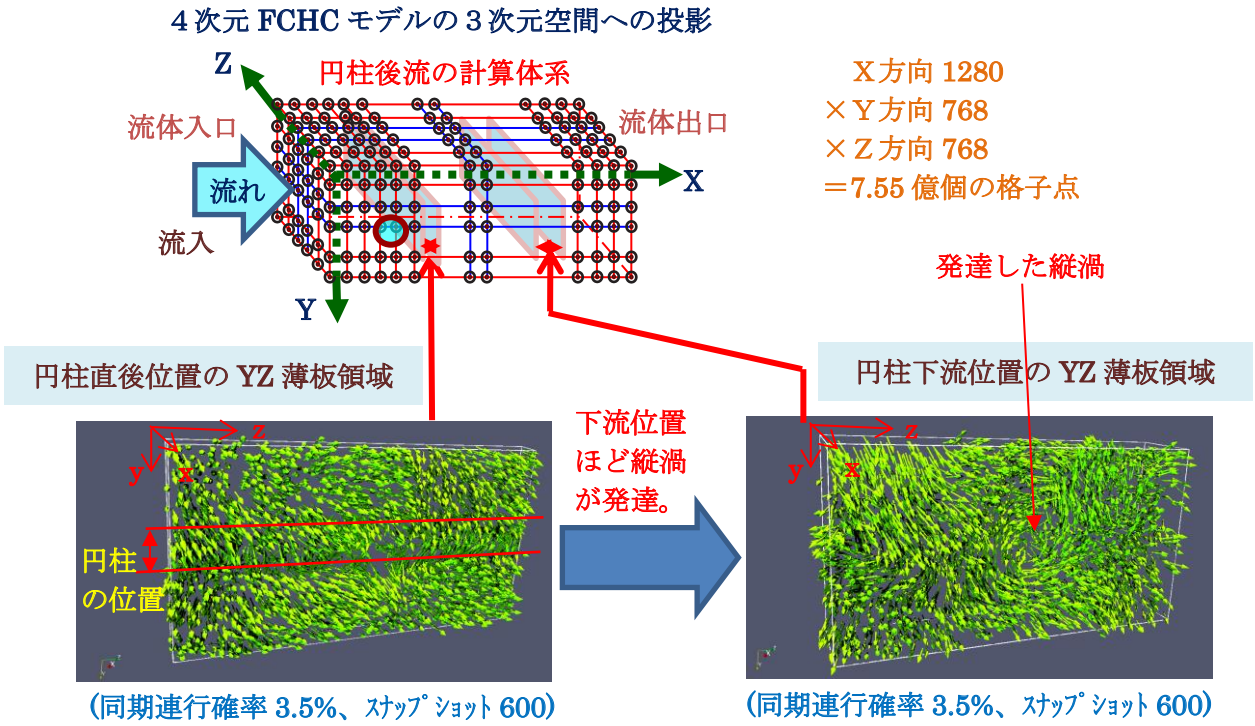


図４．“仮想反粒子場モデル”に基づく粘性制御手法による縦渦挙動創発解析
（水平円柱に左側から流体を注入し、円柱の直後及び下流位置において
流下方向に垂直な薄板領域内の運動量ベクトルを出口側から見た図）

“縦渦”は、時として非常に遠方にまで伝わり、ものづくり流体工学設計では、重要な関心事になっている。航空機の主翼両端から発生する縦渦は燃料の過剰消費につながり、船舶の船首で発生した１対の縦渦は船尾にあるプロペラに到達して損傷の原因になる。また、洋上風力発電ファームでは、風上側の風車から発生する縦渦が、風下側の風車に影響を与え、機械的な振動や発電量に悪影響を与える。

他方、風力発電機のロータブレードを円柱形にして強度上有利にできるなどのメリットがある“円柱翼風車”の研究[3]も進められている。これは、回転する円柱翼とその後ろで翼にクロスするような配置で設置された円環状の平板等の間から発生する“縦渦”によって生じる定常揚力を利用するものである。“縦渦”の発生をリスクと捉えずに、積極的に利用する点で興味深い。
このような関心もあり、手始めに、静止した水平円柱とその下流に十字交差配置された帯状の垂直な平板から発生する縦渦の計算を行ってみた。**図５**に、これを真上から見た時の流体運動量ベクトルの分布を示す。工学的に関心のある“ネックレス渦”の発生状態を模擬できている。

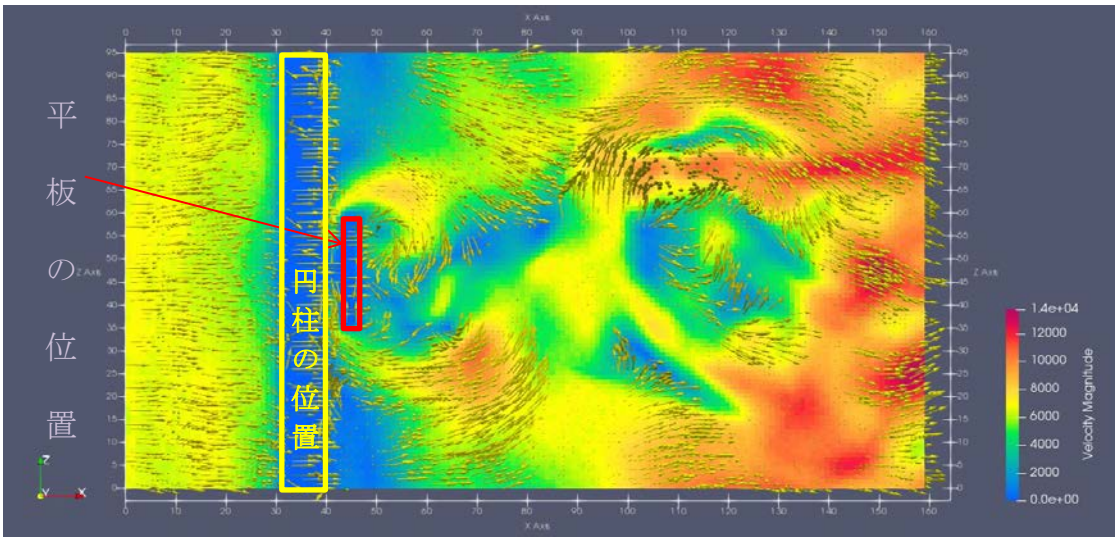


図５．円柱とその下流に十字交差配置された帯状の平板から発生する縦渦の計算事例
（同期連行確率 3.0%、ｽﾅｯﾌﾟｼｮｯﾄ 575。水平円柱の背後の位置(赤枠)に帯状の平板が垂直に配置。）

図６は、加藤らが、小型水流実験装置によって行った先行研究[4]の実験例である。円柱とその下流に十字交差配置された帯状の平板から、ある特定の条件下で発生する“ネックレス渦”を可視化している。**図５**の数値計算結果は、**図６**と似たような様相の描写に成功している。

以上は、すべて定性的な比較である。しかし、“個々の仮想粒子の動きに対して、“マクスウェルの魔”のように人為的ではあるが簡単で要素還元的な操作（達人操作)”を行うだけで、高レイノルズ数領域において発生する縦渦挙動の特徴的な様相を再現できた。このことから、これまでに開発してきた粘性制御手法は、機構論的な観点から、自然界に存在する乱流のある特性をうまく模擬できているものだと考えられる。

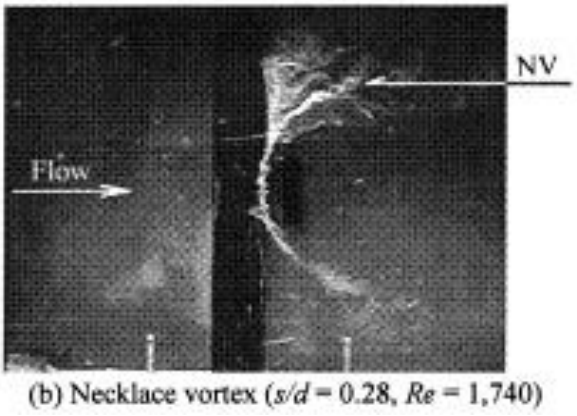


図６．【加藤らによる先行研究[4]からの引用】
円柱とその下流に十字交差配置された
帯状の平板から発生する縦渦の実験写真

もちろん、ここで用いた粘性制御手法は、“単に、似たような流体挙動を作り出しているに過ぎず、現実とは別物である”という疑問も当然ある。この点については、ここで用いた粘性制御手法が仮想粒子の挙動に与える影響を理論的な式で表現してナビエ・ストークス方程式の形に変形し、粘性項を比較することによって、当該粘性制御モデルが内包している機構論的な調整パラメータ(同期連行確率)と粘性係数の関係を式で示すことが一番わかりやすい。しかし、筆者は、前述の【“仮想光子場モデル”の発想】で述べた「非平衡統計力学にいう“揺動力”」の観点からは、「たまたま同じ向きの揺動力を続けて２回与える確率を制御する」という粘性制御手法は、単純明快でわかりやすいように思える。今後の方向としては、現実の流れを観測して、そのフィードバックにより機構論的な調整パラメータ(同期連行確率)を制御する方法を確立したい。また、フィードバック制御を導入した場合、粘性制御モデル自体がナビエ・ストークス方程式を正確に反映していなくても、流体挙動の因果関係の特徴を機構論的に正しくとらえていれば、調整パラメータの学習だけでも高精度な流体挙動の模擬が可能なこともあり得ると期待している。

３．“仮想光子場モデル”に基づく粘性制御手法による円柱後流のカルマン渦列解析

次に、“仮想光子場モデル”に基づく粘性制御を利用した円柱後流のカルマン渦列解析の事例について述べる。計算手順の大枠は、前述した**図３**と全く同じである。ただし、【同期連行過程における計算手順】においては、“仮想反粒子”は登場しない。“仮想光子”と“静止仮想粒子”の合体分離が主役になる。具体的には、以下のとおりである。

【同期連行過程における計算手順（“仮想光子場モデル”の場合)】

１．時刻ｔ－1にD向きまたは逆D向きの速度をもって格子点から出発した仮想粒子、及び、時刻ｔにD向きまたは逆D向きの速度をもって格子点に到着した仮想粒子の、合計４つの仮想粒子にだけ注目して、その存否（存在するか否か？）を調べる。

２．４つの仮想粒子のうち、時刻ｔ－1 に向きDの速度をもって出発した仮想粒子だけが存在すれば、

① 時刻ｔにも向きDの速度をもつ仮想粒子を出発させるよう努める。
⇒このため、時刻ｔにおける衝突散乱の結果、向きDに出発する仮想粒子が既に準備されていれば何もしない。準備されていない場合は、“静止仮想粒子”が存在する場合に限り、背景の仮想光子場から向きDの速度をもつ仮想光子を拾いあげて合体させ、向きDに出発できる仮想粒子を準備する。

② 時刻ｔに逆D向きに仮想粒子を出発させないよう努める。
⇒このため、時刻ｔにおける衝突散乱の結果、逆向きDに出発する仮想粒子が準備されていなければ何もしない。準備されている場合は、“静止仮想粒子”が存在しない場合に限り、向きDの速度をもつ仮想粒子を“静止仮想粒子”と“仮想光子”に分離して、逆D向きに出発できる仮想粒子を削除する。

３．２の操作は、逆D向きに対しても同じに実行する。

４．１から３の操作は、確率的に行うとともに、例えば「必ず交互に行う」などの方法により、平衡が成立するように実行する。

プログラムは省略するが、“仮想反粒子場”の場合のプログラムにおいて、“仮想粒子”と“仮想反粒子”の存否関係を、“静止仮想粒子”と“仮想光子”の存否関係に置き換えるだけなので、ビット演算としての計算手順は同じである。

以上の計算手順により、下記の条件で、円柱後流のカルマン渦列が生成する過渡変化シミュレーションを行った。シミュレーション体系の形状は、**図４**に示すものと同様である。

[過渡変化シミュレーションの条件]

シミュレーション計算を開始する最初の時刻ステップ０の時点で、各格子点には、そこに存在できる仮想粒子の最大数の 20%の数の仮想粒子をランダムな向きに配置する。この結果、疎視化して得られるマクロな運動量はゼロであり、流体は、直方体形状の中で静止している。次に、時刻ステップ１の時点から、＋X向きの運動量をもつ仮想粒子をＸ＝０の位置から注入していく。すると、時刻ステップが進むにつれて、流体全体が＋X 向きのマクロな運動量をもつようになる。このとき、＋X側の先にある直方体出口においては、出口直前に存在する格子点上の仮想粒子配置を、出口直後に存在する格子点の仮想粒子配置にコピーして、出口におけるマクロな運動量勾配がゼロになるという境界条件を近似的に実現した。また、±Y方向と±Z方向には、周期的境界条件を適用した。そして、この流れの中の入り口に近い位置に“Z方向の中心軸をもつ無限大の長さの円柱”を置き、その後流に生じる流体挙動を計算した。

計算体系における格子点配置は、
Ｘ方向 1280
×Ｙ方向　768
×Ｚ方向　192
≒1.89 億個の格子点
である。
なお、４次元ＦＣＨＣモデルであるため、上記の格子点配置に加えて、同じ３次元位置に４個ずつ第４の方向に沿って格子点を配置している、
また、０から 51200 時刻ステップまでの計算を行い、256 時刻ステップ経過するたびに一つのスナップショットを出力させた。スナップショットの総数は、201 枚である。
スナップショットは、円柱軸を通りＹ軸に垂直な平面上における流体の運動量ベクトルである。
疎視化は、４×４×４の３次元格子点領域ごとに行っている。
可視化には、ParaView を用いた。

図７に示したスナップショットは、いずれも、過渡変化が十分安定化した後の、195 枚目のスナップショットの瞬間画像である。
“同期連行確率”を０から増やしていくと、カルマン渦列の挙動が乱流化していくのがわかる。実際は、201 枚のスナップショットの動画を ParaView で観察できるので、この変化は一層明らかである。

本計算には、東北大学サイバーサイエンスセンターのＡＯＢＡ－Ａを使用し、１ケース（上述のとおり、1.89 億個の 3 次元格子点についての時間発展計算を 51200 ステップ行う計算）を、「８コア×16CPU」の mpi による 128 並列計算で約 110 分であった。（ベクトル化率：約 97.5%、ベクトル長：約 200）
なお、本プログラムは日々大きな改良をしているため、特別なチューニングは行っていない。

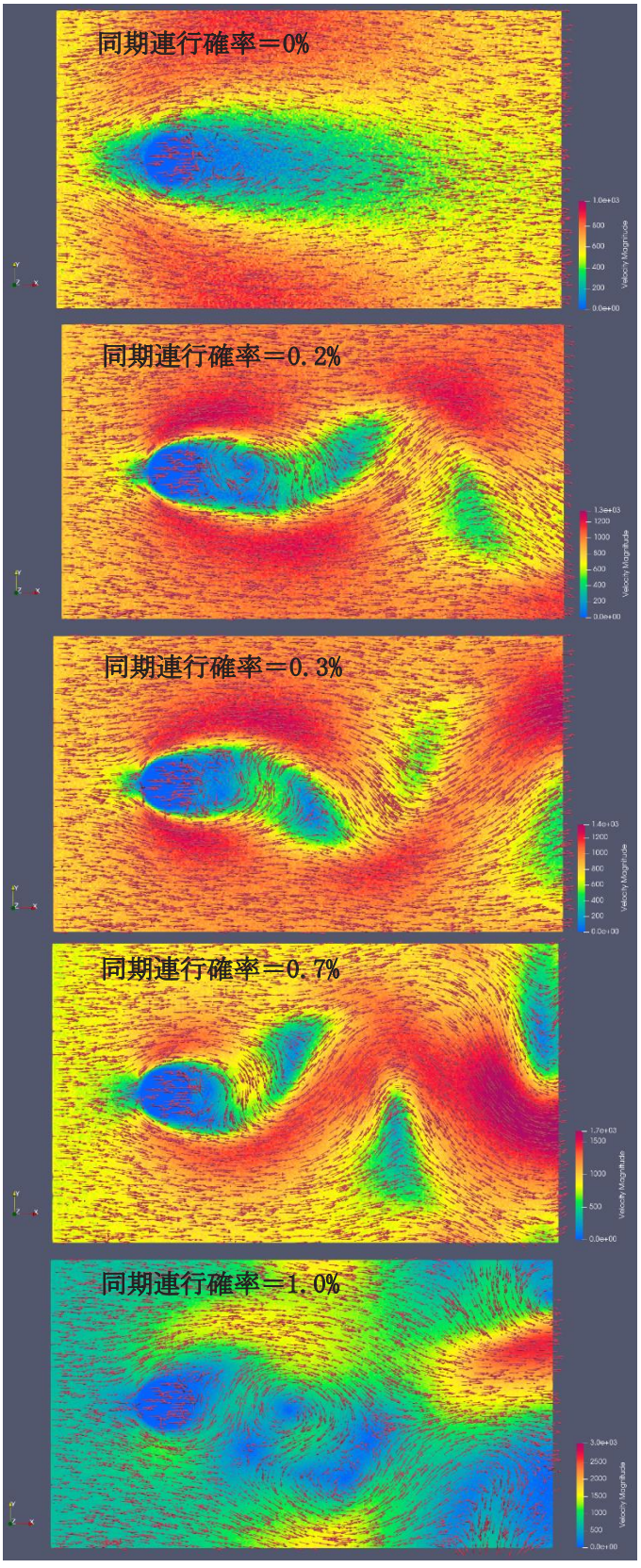


図７．“仮想光子モデル”に基づく粘性制御手法による円柱後流のカルマン渦列解析

４.“格子ガス法”の将来性

筆者が行ってきた今回の３か年の研究は、「流れの中に置かれた物体境界やその直後におけるせん断流の“局所スケール領域”」から、「物体から相当離れた後流までの“大規模スケール領域”」までの縦渦挙動を、“格子ガス法”の各格子点における仮想粒子挙動の簡単な計算規則から自己組織化的に再現(創発)できることを示すことを主要な目的にしている。

今後は、この計算手法を高解像度な流体工学設計の実現に役立てる方法を追求していきたい。筆者が、このような流体工学設計の実現に向けて、なぜ、あえて“格子ガス法サバイバル”の道に期待するのか？その理由は以下のとおりである。

（１）“格子ガス法”における基本的問題の解決：

① 等方性の問題、ガリレイ不変性の問題は、Teixeira の FCHC モデル[5]によって解決済み。

② 例えば、Teixeira による「３速さ 54 速度モデル」では、本手法による解析結果を、ナビエ・ストークス方程式を解く標準的な数値流体力学(非圧縮性流体)の結果と比較した場合、マッハ数に関する３次の精度まで一致することが Teixeira の論文[5]において証明済み。

③ また、時間発展計算式の線形性(multi linear)により、計算された運動量のノイズがどんなに大きく変動しても、その平均値は常に正しい結果に収束することも、同論文において証明済み。(cf. Shot Noise Theorem：[5]の第６章)

（２）マルチスケール解析実現に向けた将来性：

① １格子点に関する時間発展計算を１ビット幅で行うことができるので、非常に多数の格子点に関する演算を高効率な超並列計算で実行可能。

② 仮想粒子の“衝突散乱”、“並進移動”等の時間発展計算をすべて「ビット演算」で実行できるので、時間発展計算での誤差の蓄積がない。特に、どんなに激しい流れに対しても、安定的かつ効率的に結果を得ることができるので、乱流解析に好都合。

③ Bosch による「仮想反粒子を利用した“マルチグリッドアルゴリズム”[6]」によって、流体中の物体近傍など、詳細な解析が必要な領域にだけ細かい格子の割り当てが可能。

④ ゲート型量子コンピュータは、格子ガス法が行う“確率的なビット演算”に向いており、これを利用した高速計算が近々実現できる可能性がある。

（３）マルチフィジックス解析実現に向けた将来性：

① 二相流[7]、高温物体による流体の蒸気爆発[8]など様々な流体関連現象について、格子ガス法による解析事例が蓄積。

② 粒子の挙動を追跡するので、化学反応や燃焼現象を直接的なモデル化が容易。

【注】格子ガスオートマトン法のマルチフィジックス解析への適用事例は、少数の例示では表現できないほど数多く存在している。

（４）リアルワールド解析実現に向けた将来性：

① 格子ガス法の時間発展計算はすべてリカレント型ニューラルネットワークで表現できる[9、10]ので、ニューラルネットの分野で開発された学習アルゴリズムを計測融合シミュレーション（リアルタイムデータ同化）に直接利用することが可能。

② リアルタイムデータ同化が可能になれば、各種工学システムの IoT によるエッジコンピューティング制御のニーズに対応可能。

③ 計算過程をリカレント型ニューラルネットワークで表現できることは、深層学習により簡単なニューラルネットワークに変換することで低計算量の「サロゲートモデル」を作ることが可能。

以上。

謝辞

本研究で実行したシミュレーション計算には、本文にも述べたとおり、すべて、東北大学サイバーサイエンスセンターのベクトル型スーパーコンピュータ AOBA-A(SX-Aurora TSUBASA)の「８コア×16CPU」を利用した。また、利用にあたって同センター関係各位のご親切なご指導とご協力をいただき、心から感謝する次第である。今後とも、このような使い勝手のよいベクトル型スーパーコンピュータのさらなる開発導入と同センターの有意義な活動の継続を期待している。特に、昨年 AOBA-S が導入されているので、今後は、この利用も試みたい。

参考文献

[1] 香取,“非平衡統計力学”, pp89-127, 1999, 裳華房 (ISBN 978-4-7853-2086-7)
[2] 松岡,“リカレント型ビット演算による多様な縦渦挙動の創発”, SENAC Vol.56 No.1, pp.24-36, 2023
[3] 坂本, Hemsuwan, 高橋,“縦渦の定常揚力により駆動する円柱翼風車の抗力特性”, 日本機械学会論文集, Vol.87, No.894, 2021
[4] 加藤, 小出, 高橋, 白樫,“円柱の下流に十字交差配置された帯状の平板による振動制御”（第１報,固定系からの縦渦流出）,日本機械学会論文集（B 編）, 73 巻 728 号(2007-4), pp.957-964, 論文 No.06-0029
[5] Christopher M. Teixeira, “Continuum Limit of Lattice Gas Fluid Dynamics”, Ph.D. Thesis, MIT, 1993
[6] Robert P. Bosch, Jr.,“A Multigrid Algorithm for Lattice Gases”, MIT, 1993
[7]妻屋彰,「格子ガスオートマトンを用いた二相流の機構論的解析」, 学位論文, 東京大学, 1998
[8]栃尾大輔,「蒸気爆発現象のトリガリング過程における蒸気幕崩壊現象に関する研究」, 学位論文, 筑波大学, 2003
[9] 松岡, 菊池,“リカレントニューラルネットワークによる実世界流れ場解析用時間発展計算モデルの探求”, SENAC Vol.53 No.1, pp.25-33, 2020
[10] 松岡,“格子ガス法流体解析モデルとニューラルネットワークの融合”, SENAC Vol.54 No.1, pp.39-49, 2021

［大規模科学計算システム］【AOBA-S の利用法】

サブシステム AOBA-S の利用法

情報部デジタルサービス支援課

1　はじめに

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

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

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

2　システムの構成

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

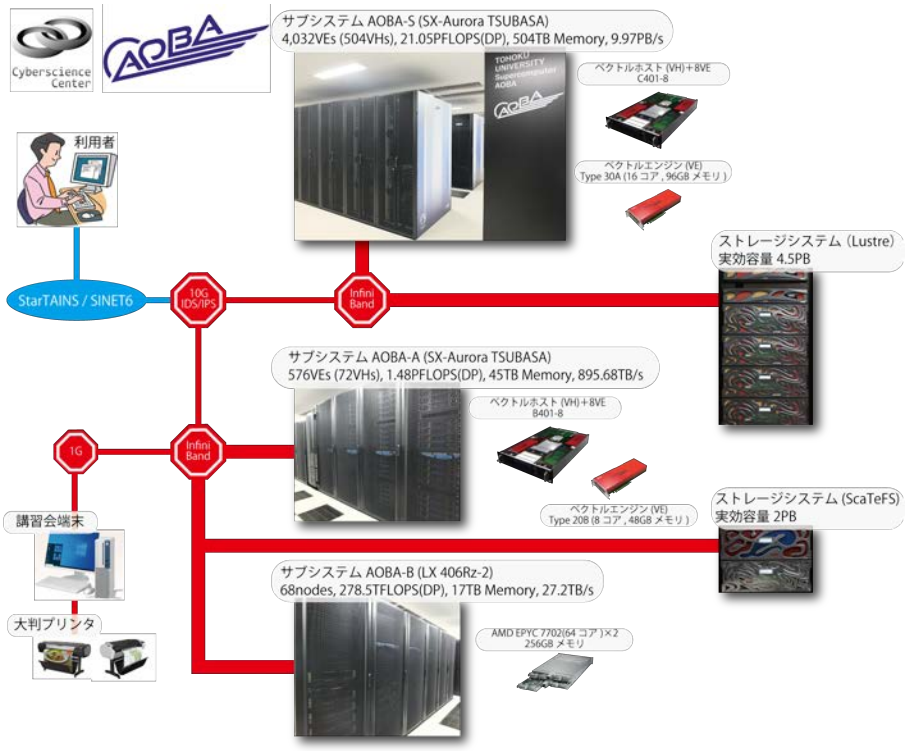


図 1　スーパーコンピュータ AOBA の構成

2.1　サブシステム AOBA-S の特徴

2.1.1　SX-Aurora TSUBASA アーキテクチャ

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

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

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

図 2　サブシステム AOBA-S（32 ラック）

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

2.1.2　ベクトルプロセッサ

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

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

2.1.3 ソフトウェア

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

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

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

2.2 プログラムの実行方法

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

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

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

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

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

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

表 1 実行の種類、特徴、最大並列数、最大メモリ量

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

3　利用者向けサーバへのログイン方法

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

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

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

本センターのシステムをはじめて利用する方は以下の手続きが必要です。

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

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

3.1　利用者向けサーバ名と用途

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

表 2　AOBA-S 用の利用者向けサーバ

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

3.2 SSH 認証鍵ペアの作成とログインまでの概要

図 3 に、鍵ペアの作成からログインまでの流れを示します。
1. 鍵ペアの作成は 3.3 節で
2. ターミナルソフトの設定および 3. ログイン については 3.4 節で

それぞれ解説します。

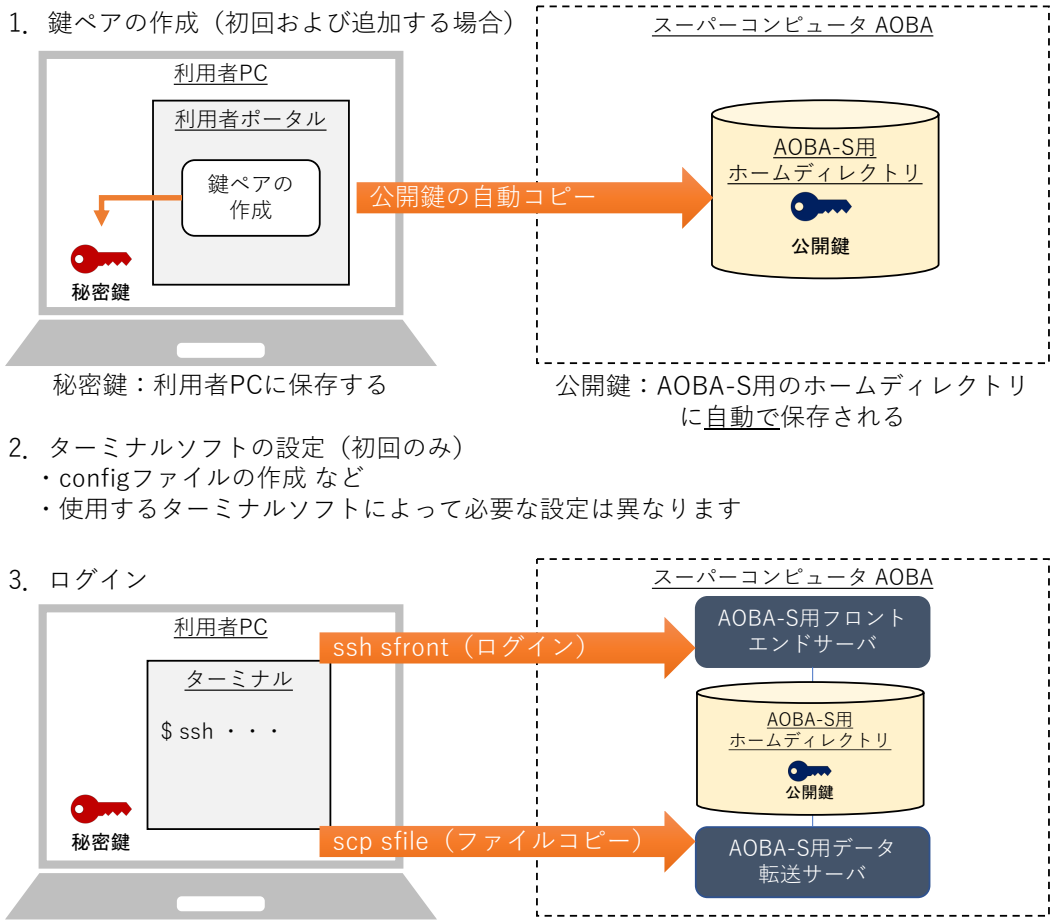


図 3　鍵ペアの作成からログインまで

1. 鍵ペアの作成（初回ログイン時、およびログイン端末を追加する場合）

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

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

2. ターミナルソフトの設定（初回ログイン時）

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

3. ログイン

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

3.3　公開鍵認証方式で使用する鍵ペアの作成

3.3.1　公開鍵認証方式を使用する上での注意事項

以下の注意事項を必ず守ってください。

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

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

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

■初回ログイン時　鍵ペアの作成は利用者ポータルで行います。

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

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

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

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

3.4　利用者向けサーバへのログイン方法

3.4.1　ターミナルソフトの設定（各 PC での設定）

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

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

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

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

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

リスト 1　秘密鍵のパーミッション変更

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

以降は Windows、macOS ／ Linux 共通です。

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

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

リスト 2　config ファイルの設定方法

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

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

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

3.4.2　AOBA-S 用フロントエンドサーバへのログイン

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

リスト 3　AOBA-S 用フロントエンドサーバへのログイン

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

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

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

3.4.3　AOBA-S 用データ転送サーバの利用方法

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

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

リスト 4　AOBA-S 用データ転送サーバ

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

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

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

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

リスト 5　AOBA-S 用データ転送サーバ

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

AOBA-S 用データ転送サーバ上でのマウントポイントは以下の通りです。

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

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

リスト 6　AOBA-S 用データ転送サーバ

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

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

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

3.4.4　HPCI 用フロントエンドサーバへのログイン

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

リスト 7　HPCI 用フロントエンドサーバ

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

3.5 ログインシェルの確認と変更

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

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

リスト 8　ログインシェルの確認と変更

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

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

3.6 LDAP パスワードの変更

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

(1) 以下の URL から利用者ポータルにログインします。利用者ポータルには、利用者番号と現在の LDAP パスワードでログインします。
【利用者ポータル】https://www.ss.cc.tohoku.ac.jp/portal/

(2)「パスワード変更」ボタンをクリックします。

(3) 利用者ポータルの画面の説明に従い、新しい LDAP パスワードを設定します。

(4) 以下で使用するパスワードが変更されます。

- 利用者ポータルへのログイン
- 大判カラープリンタのプリンタサーバへのログイン
- ログインシェルの変更時

4　プログラムのコンパイルと実行

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

4.1　AOBA-S 用フロントエンドサーバへのログイン

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

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

リスト 9　AOBA-S 用フロントエンドサーバへのログイン

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

4.2　ソースコードの作成

ソースコードを作成する代表的な方法は以下の通りです。

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

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

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

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

リスト 10　nkf コマンドの使用例

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

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

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

4.3 コンパイル

4.3.1 コンパイラの仕様

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

Fortran コンパイラ

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

C/C++ コンパイラ

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

4.3.2 コンパイルの手順

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

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

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

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

表 3　Fotran,C/C++ コンパイラの主なオプション

コンパイルオプション	機能
-On	最適化レベルを指定する
n に指定できる値	
4	言語仕様を逸脱した副作用を伴う最大限の最適化・自動ベクトル化を適用する
3	副作用を伴う最適化・自動ベクトル化、および、多重ループの最適化を適用する
2	（既定値）副作用を伴う最適化・自動ベクトル化を適用する
1	副作用を伴わない最適化・自動ベクトル化を適用する
0	最適化、自動ベクトル化、並列化、インライン展開を適用しない （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ください）
-finline-functions	自動インライン展開機能を適用する
-mparallel	自動並列化機能を利用する
-fopenmp	OpenMP 並列化を利用する
-mno-parallel-omp-routine	OpenMP ディレクティブを含むルーチンを自動並列化しない （-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
-report-all	コード生成リスト、診断メッセージリスト、編集リスト、インラインリスト、オプション リスト、ベクトルリストを出力する
-fcheck=bounds	バウンスチェック（配列の上下限のチェック）を行う
-fcheck=all	仮引数のエリアスへの代入、ビット intrinsic な引数、配列の上下限、不正なポインタ、DO ループのステップ値が 0 かどうか、整数オーバーフロー、ポインタ参照、省略可能な引数の参照、不正な再帰呼出しのチェックを行う

リスト 11　nfort コマンドの使用例

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

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

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

リスト 12　mpinfort コマンドの使用例

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

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

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

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

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

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

リスト 13 ncc/nc++ コマンドの使用例

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

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

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

リスト 14 mpincc/mpinc++ コマンドの使用例

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

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

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

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

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

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

図 4　Ftrace Viewer の表示例

4.4　バッチリクエストによるジョブの実行

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

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

4.4.1　バッチリクエストの概要

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

図 5 にバッチリクエスト実行の概念図を示します。

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

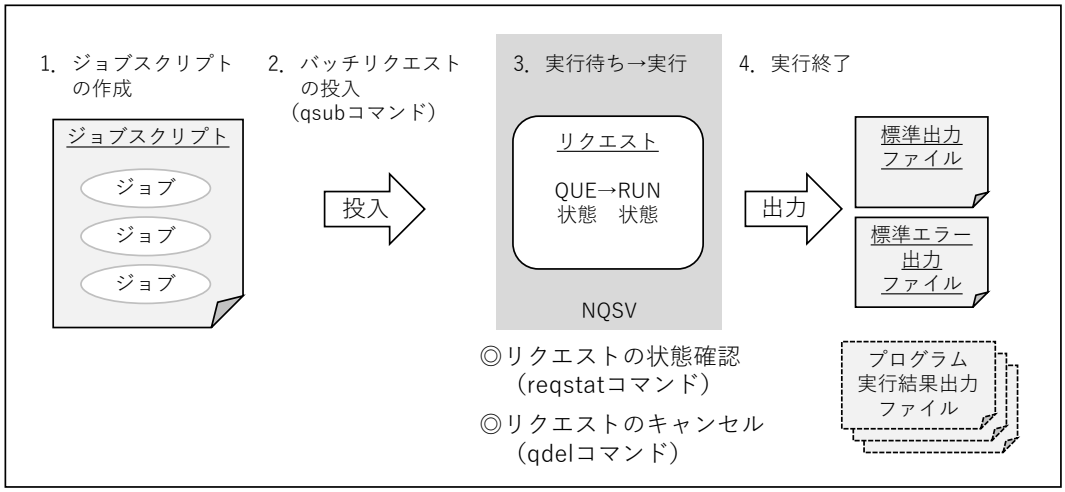


図 5　バッチリクエストの概念図

1. ジョブスクリプトの作成

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

2. バッチリクエストの投入
3. 実行待ち→実行

以下のコマンドでバッチリクエストを操作します。

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

4. 実行終了

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

終了したリクエストは reqstat コマンドの表示から削除されます。

4.4.2 キュー構成

リクエストの投入キューについて説明します。

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

表 4　サブシステム AOBA-S のキュー構成

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

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

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

4.4.3 バッチリクエストの作成

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

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

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

4.4.4　ジョブスクリプトの例

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

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

リスト 15　ジョブスクリプトの例（逐次実行）

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

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

リスト 16　ジョブスクリプトの例（逐次実行、無料キュー）

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

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

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

リスト 17　ジョブスクリプトの例（自動並列化/OpenMP 並列実行）

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

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

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

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

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

リスト 18　ジョブスクリプトの例（MPI 並列実行）

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

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

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

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

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

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

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

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

リスト 20　出力をプロセス毎に分割する方法

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

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

表 5　NMPI_SEPSELECT の引数と出力形式

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

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

4.4.5　qsub コマンドの主なオプション

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

表 6　qsub コマンドの主なオプションと機能

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

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

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

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

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

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

4.4.6　バッチリクエストの投入

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

リスト 21　qsub コマンドによるバッチリクエスト投入

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

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

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

リスト 22　qsub コマンドによるバッチリクエスト投入

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

4.4.7　バッチリクエストの実行

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

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

4.4.8　バッチリクエストの状態確認

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

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

表 7　reqstat コマンドの表示項目

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

表 8　reqstat コマンドの表示項目

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

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

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

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

4.4.9　バッチリクエストのキャンセル

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

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

リクエストがキャンセルされるとメッセージが表示されます。

リスト 23　qdel コマンド

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

4.4.10　バッチリクエストの終了

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

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

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

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

出すようにしてください。

4.5 会話リクエストの投入と利用

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

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

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

4.5.1 qlogin による会話リクエストの投入方法

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

リスト 24 qlogin コマンド

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

リスト 25 に ve-gdb（VE 用 GDB）の起動方法を示します。

リスト 25 ve-gdb コマンド

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

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

5　SX-Aurora TSUBASA 用数値演算ライブラリ

5.1　NLC (NEC Numeric Library Collection)

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

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

5.1.1　Fortran プログラムから利用する場合

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

表 9　Fortran 用 NLC の機能概要

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

5.1.2　C プログラムから利用する場合

C プログラムから利用する場合、表 10 に示すライブラリが使用できます。

表 10　C 言語用 NLC の機能概要

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

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

6 マニュアル

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

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

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

6.1 コンパイラマニュアル

コンパイラについては以下のマニュアルをご参照ください。

■SDK

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

■NEC MPI

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

6.2 性能情報取得についてのマニュアル

実行時の性能情報取得については以下のマニュアルをご参照ください。

■SDK

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

6.3 NQSV についてのマニュアル

NQSV の利用方法については以下のマニュアルをご参照ください。

■NQSV

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

6.4 GDB についてのマニュアル

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

■VEOS

- GDB の相違点

6.5　数値計算ライブラリ（NLC）マニュアル

数値計算ライブラリ（NLC）については以下のマニュアルをご参照ください。

■SDK

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

7　おわりに

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

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

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

センターから運用に関するお知らせは以下をご参照ください。

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

［大規模科学計算システム］【AOBA-A および AOBA-B の利用法】

鍵ペアの作成とログイン方法

情報部デジタルサービス支援課

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。

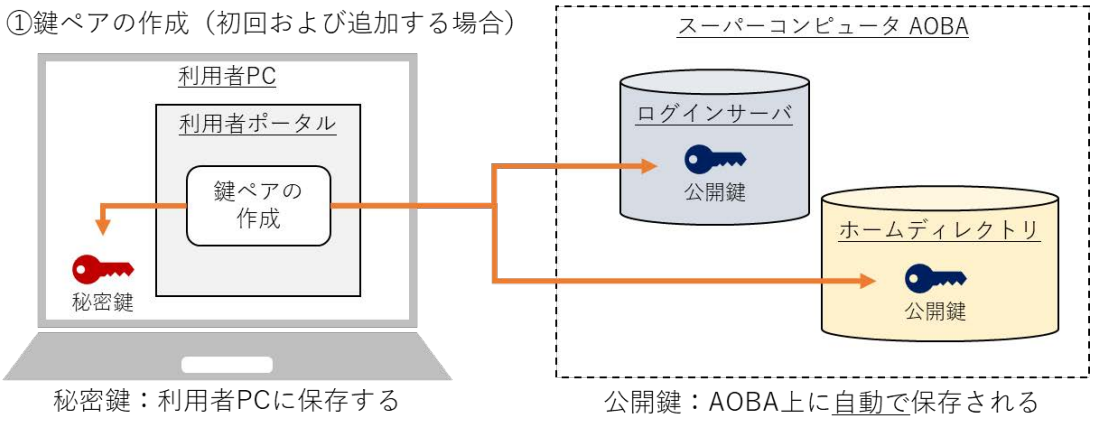


図 1　鍵ペア作成からログインまでの流れ

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

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］【AOBA-A および AOBA-B の利用法】

サブシステム AOBA-A の利用法

情報部デジタルサービス支援課

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。

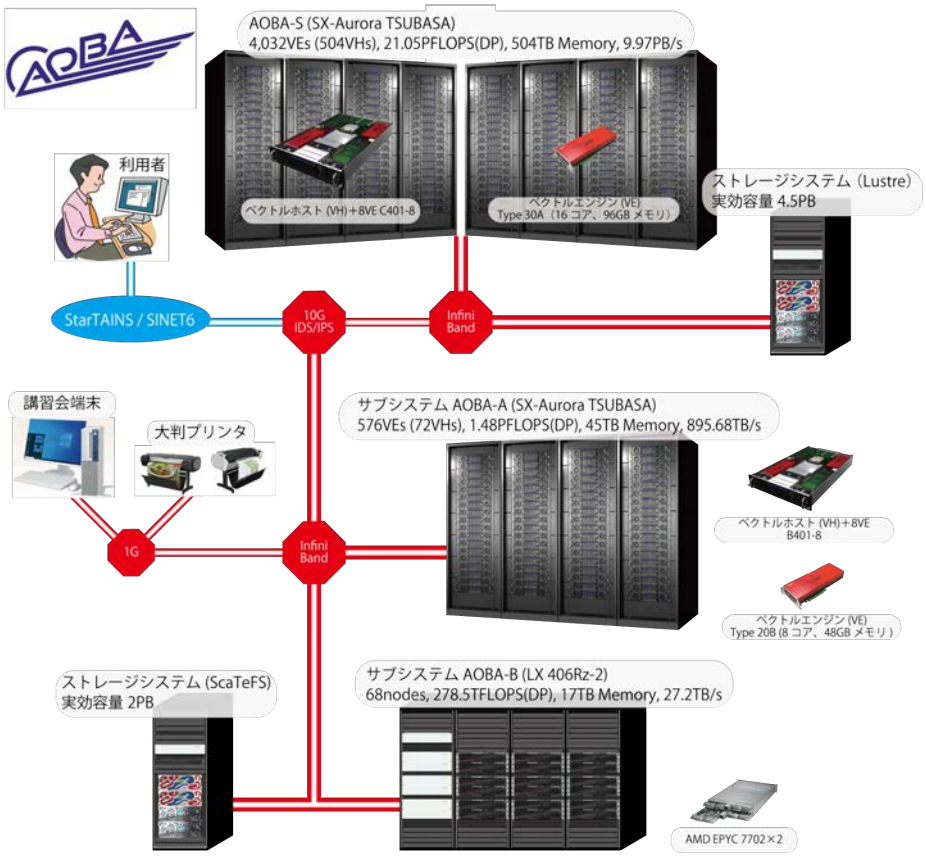


図 1　スーパーコンピュータ AOBA の構成

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。

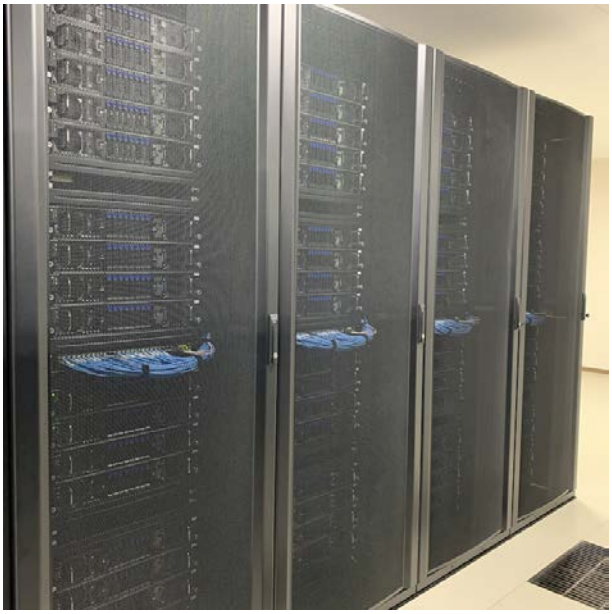


図 2 サブシステム AOBA-A（4 ラック）

- 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。

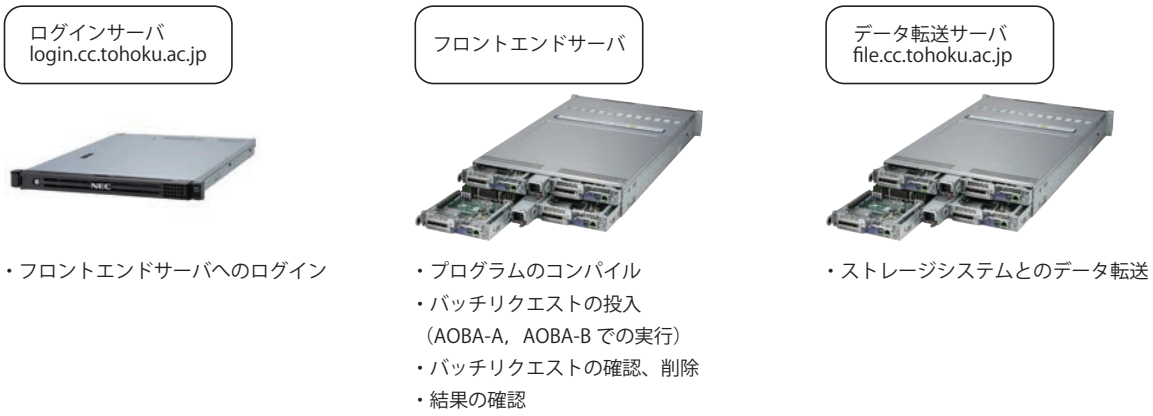


図 3　ログインサーバ，フロントエンドサーバ，データ転送サーバ

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。

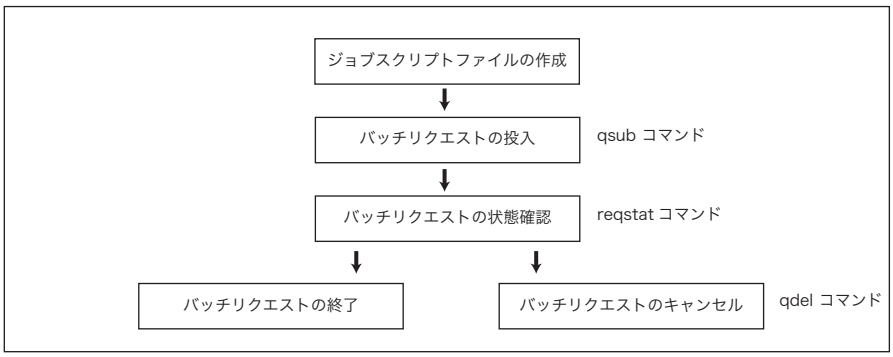


図 5　バッチリクエストの概念図

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 sxf のどちらか 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）

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。

サブシステム AOBA-B の利用法

情報部デジタルサービス支援課

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。

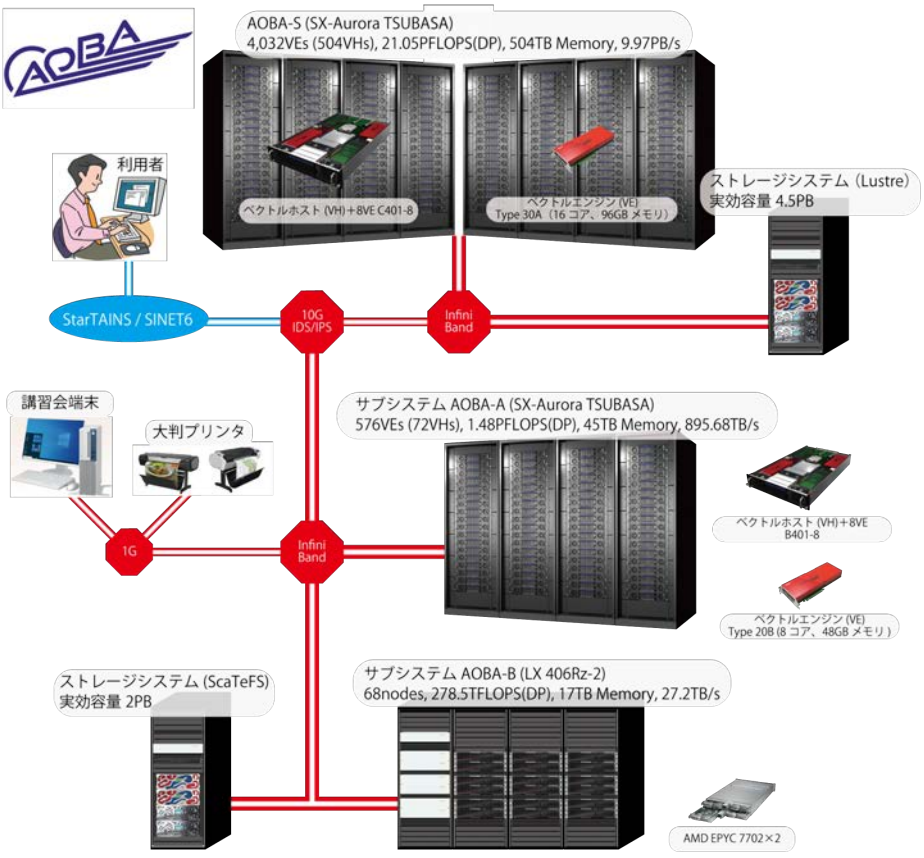


図 1　スーパーコンピュータ AOBA の構成

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。

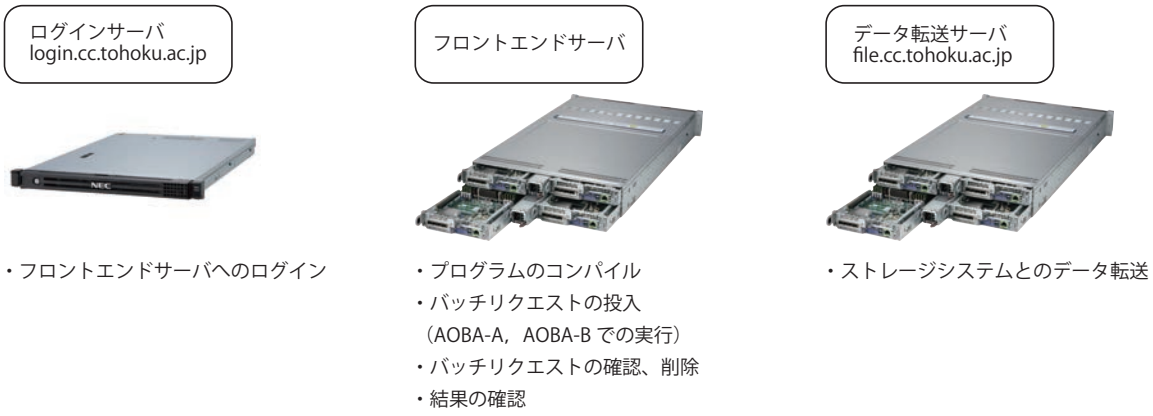


図 3 ログインサーバ，フロントエンドサーバ，データ転送サーバ

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。

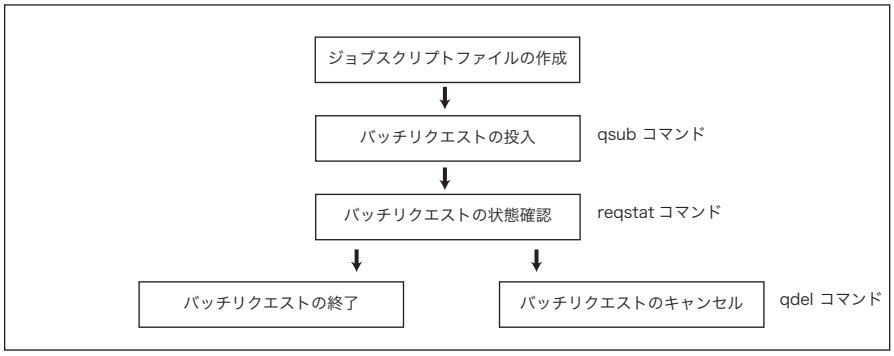


図 4　バッチリクエストの概念図

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］【AOBA-A および AOBA-B の利用法】

ストレージシステムの利用法

情報部デジタルサービス支援課

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．

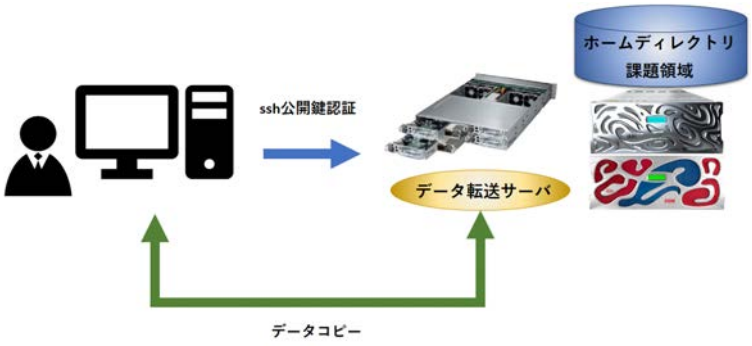


図 1　アクセスイメージ

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 -i 秘密鍵ファイル ローカル PC のファイル名 利用者番号@file.cc.tohoku.ac.jp:リモートの保存先パス

  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 -i ~/.ssh/id_rsa_cc 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 -i 秘密鍵ファイル 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 -i ~/.ssh/id_rsa_cc 利用者番号@file.cc.tohoku.ac.jp:sample.txt ./

  詳しい用例については man コマンド等を利用し，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 -i 秘密鍵ファイル 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 -i 秘密鍵ファイル 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 コマンド等を利用し，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．

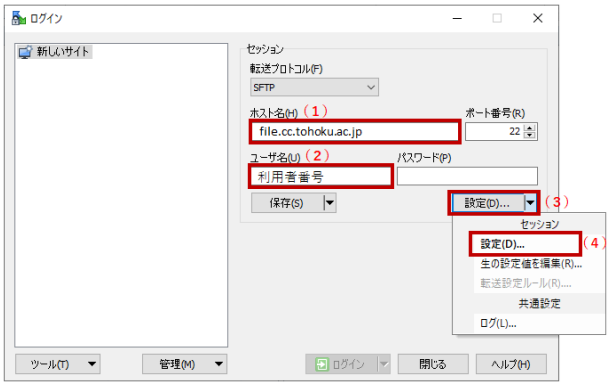


図 2　WinSCP 設定画面

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.

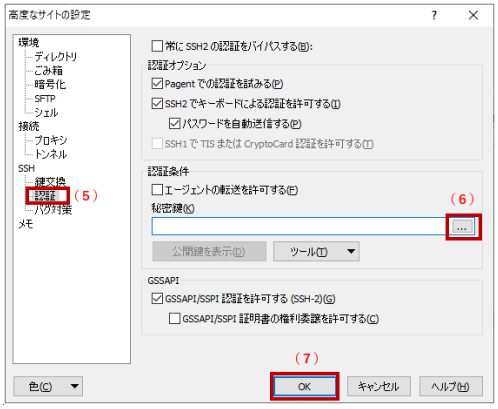


図 3　WinSCP 鍵認証設定画面

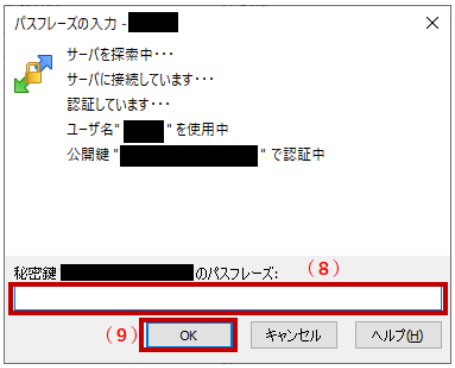


図 4　WinSCP 鍵認証画面

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（上図（8））した上で， OK （上図（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.

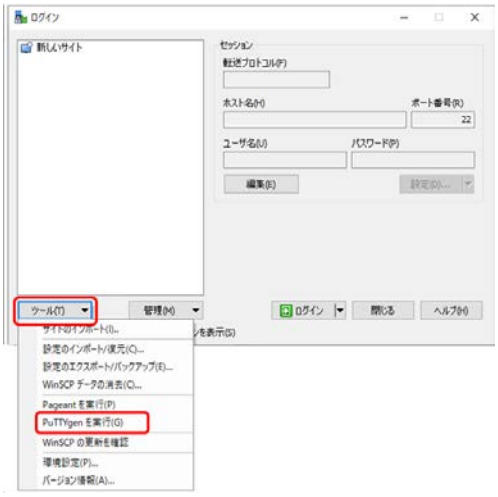


図 5　WinSCP 初期設定画面

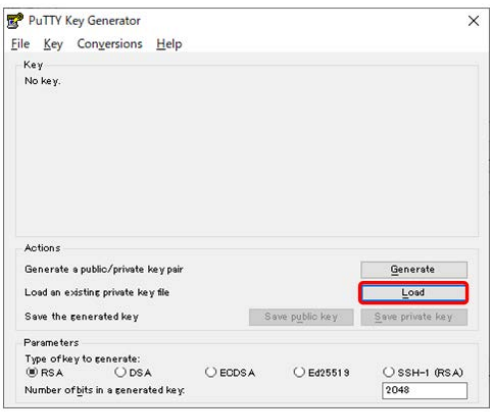


図 6　PuTTY Key Generator 画面

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.

［報　告］

令和 5 年度 サイバーサイエンスセンター講習会報告

No.	講習会名 開催方式	開催日時	受講者数 ( ) 内はオンライン参加者数	講師	内容
1	はじめての Linux ハイブリッド型	5 月 23 日(火) 13:00-15:00	3 (3)	小野 (情報部デジタルサービス支援課)	・Linux システムの基本的な使い方 ・エディタの使い方
2	はじめてのスパコン ハイブリッド型	5 月 25 日(木) 13:00-15:00	4 (2)	山下 (情報部デジタルサービス支援課)	・スーパーコンピュータの紹介と利用法入門
3	はじめての並列化 オンライン	5 月 31 日(水) 13:00-15:00	2 (2)	小松 (サイバーサイエンスセンター)	・並列プログラミングの概要
4	MATLAB 入門 対面	6 月 30 日(金) 13:00-16:30	2	陳 (秋田県立大学)	・MATLAB の基本的な使い方
5	ネットワークとセキュリティ入門 オンライン	8 月 3 日(木) 13:30-15:30	12	水木 (サイバーサイエンスセンター)	・ネットワークの基本的な仕組み ・ネットワークの危険性と安全対策
6	Gaussian 入門 対面	8 月 23 日(水) 13:00-16:30	3	岸本 (理学研究科)	・Gaussian の基本的な使い方
7	Fortran 入門 対面	9 月 7 日(木) ～8 日(金) 7 日 10:00-17:00 8 日 10:00-12:00	中止	田口 (日本原子力研究開発機構)	・Fortran の入門編
8	Mathematica 入門 対面	9 月 12 日(火) 13:00-16:30	中止	横井 (尚絅学院大学)	・Mathematica の基本的な使い方
9	新システムの紹介と利用法 オンライン	9 月 25 日月) 13:00-14:30	9	滝沢 (サイバーサイエンスセンター)、山下(情報部デジタルサービス支援課)	・新スーパーコンピュータ AOBA-S の紹介 ・利用方法、利用負担金について
10	はじめての Linux ハイブリッド型	9 月 27 日(水) 13:00-15:00	6 (6)	大泉 (情報部デジタルサービス支援課)	・Linux システムの基本的な使い方 ・エディタの使い方
11	並列プログラミング入門Ⅰ (OpenMP) オンライン	10 月 3 日 (火) 13:00-16:00	3 (3)	小松 (サイバーサイエンスセンター)	・並列プログラミングの概要 ・OpenMP による並列プログラミングの基礎 ・利用法
12	並列プログラミング入門Ⅱ (MPI) ハイブリッド型	10 月 10 日(火) 13:00-16:00	1 (1)	下村 (サイバーサイエンスセンター)	・MPI による並列プログラミングの基礎 ・利用法
13	SX-Aurora TSUBASA の性能分析・高速化 ハイブリッド型	10 月 17 日(火) 13:00-16:00	3 (3)	江川 (サイバーサイエンスセンター)	・スーパーコンピュータでの性能解析から最適化まで

＊ハイブリッド型講習会で、受講者数()内は全受講者のうちの対面式受講者数

［報　告］
〈計算科学・計算機科学人材育成のためのスーパーコンピュータ無償提供制度〉

東北大学統合化学国際共同大学院（GP-Chem）先進化学国際講義Ⅰにおける量子化学実習

菅野　学
東北大学大学院理学研究科化学専攻

東北大学統合化学国際共同大学院（GP-Chem）は、化学の多様な分野を統合した研究・教育を行うことを目的として2022年4月に設置されました。理学・工学・薬学・農学・情報科学・生命科学・環境科学の7研究科に所属する大学院生から選抜し、化学およびその学際領域を幅広くカバーする分野融合型の教育プログラムや、国際的なリーダーシップを養うためのメニューを通して、最先端の学術研究を牽引し国際的に活躍できる人材の育成を目指しています。

先進化学国際講義Ⅰは、化学の未探究領域における最新の進歩を紹介するコースであり、その一環として量子化学実習を開講しました。近年の理論の発展、コンピューターの性能向上、使いやすいソフトウェアの開発などによって、量子化学計算は生命・創薬・材料などを含むあらゆる分野で活用されています。普段は量子化学計算に接する機会の少ない大学院生がその基礎を学ぶことで、各自の研究に役立てるだけでなく、俯瞰的視野と実践力を養うための実習です。

2024年1月16日（火）と18日（木）にサイバーサイエンスセンターの端末機室を3講時から5講時までの時間（13:00 ～ 17:50）お借りし、留学生を含む受講生11名を2つのグループ（16日は5名、18日は6名）に振り分けて同じ内容の実習を英語で行いました。Windows端末にインストールされたGaussian社の量子化学計算ソフトウェアGaussian 16およびそれ専用の可視化ソフトウェアGaussView 6を使用し、受講生各自が以下の課題に取り組みました。

1. 分子モデル（初期構造）の作成とそれを始点とした分子構造の最適化（安定構造の探索）
2. 分子の基準振動解析（赤外吸収スペクトル計算）
3. 溶媒効果の考慮
4. 分子軌道や電子密度などの可視化
5. 化学反応経路の探索（遷移状態計算）
6. 電子励起状態の評価（紫外・可視吸収スペクトル計算）

Gaussian 16は世界で最も広く使われている量子化学計算ソフトウェアであり、それをGaussView 6から呼び出すことにより、Windows上の簡単なマウス操作で計算を実行できます。当日の実習はGaussian 16とGaussView 6の実践的な操作法の習得に重点を置き、量子化学の基礎理論に関しては受講生自身が文献等を調査して学習するレポート課題として加えました。

筆者は2020年度からサイバーサイエンスセンターのテクニカルアシスタント(Gaussian担当)を務めており、主に実験研究を専門としている学生から届いた量子化学計算の初歩に関する質問に答えることが多くあります。実験研究者にも量子化学計算の重要性が認知されていることを実感しつつ、習得の機会が少ないことも事実と思います。本実習のような取り組みが、その解決の一助となることを願っております。

本実習は、サイバーサイエンスセンターの「計算科学・計算機科学人材育成のためのスーパーコンピュータ無償提供制度」の支援を受けて実施させていただきました。関係者の皆様に深く感謝いたします。

［スーパーコンピュータ AOBA のお知らせより］

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ウェブサイトに掲載されたお知らせの一部を転載しています。
https://www.ss.cc.tohoku.ac.jp/information/

高速化推進研究活動報告 第 8 号の発行について

高速化推進研究活動報告は本センターにおける利用支援や共同研究等の活動の成果をまとめたもので、第 8 号では 2018 年度から 2022 年度までの成果が収録されています。

第 8 号は以下リンク先よりご参照ください。これまでに発行された第 1 号から第 7 号も閲覧いただけます。

https://www.ss.cc.tohoku.ac.jp/tuning-support/

（スーパーコンピューティング研究部，情報部デジタルサービス支援課）

コンパイラのバージョンアップについて

2024 年 4 月 1 日に AOBA のコンパイラをバージョンアップいたします。

システム	コンパイラ名	旧バージョン	新バージョン	ドキュメント
AOBA-S AOBA-A	Fortran Compiler	5.0.2	5.2.0	マニュアル&リリースノート
	C/C++ Compiler	5.0.2	5.2.0	
	MPI※1	3.4.0	3.6.0	
	Intel Compiler (VH 用プログラム) ※2	oneAPI 2023.2.0	oneAPI 2024.0.1	oneAPI マニュアル関連
AOBA-B	Intel Compiler ※2			
	AOCC	4.0	4.2	リリースノート
	AOCL	4.0	4.2	リリースノート

※1 MPI を利用するプログラムは再コンパイルが必要
※2 Intel oneAPI 2024.0.1 の環境変数設定ファイルは、bash 向けのみの提供

（情報部デジタルサービス支援課）

商用アプリケーションのバージョンアップについて

数値解析ソフトウェア「MATLAB」および、数式処理システム「Mathematica」のバージョンアップを行いましたのでお知らせいたします。
新機能の概要、機能の詳細については開発元 Web サイトをご参照ください。

MATLAB

- バージョン：R2024a
- バージョンアップ日：2024 年 4 月 11 日
- サービスホスト：フロントエンドサーバ、AOBA-B
- 起動コマンド：（GUI 版）matlab　（コマンドライン版）matlab -nojvm -nosplash -nodesktop -nodisplay
- 開発元 Web サイト：https://jp.mathworks.com/products/new_products/latest_features.html

Mathematica

- バージョン：14.0
- バージョンアップ日：2024 年 4 月 1 日
- サービスホスト：フロントエンドサーバ
- 起動コマンド：（GUI 版）mathematica　（コマンドライン版）math
- 開発元 Web サイト：https://www.wolfram.com/mathematica/new-in-14/

（情報部デジタルサービス支援課）

令和 6 年度共同研究課題採択結果（第 1 回）について

本センターでは、大規模科学計算システムの利用者と共同でプログラムやアルゴリズムを開発する共同研究を行っています。令和 6 年度の第 1 回募集に応募されたものについて共同研究専門部会で審査の結果、14 件が採択されましたのでお知らせします。詳細は以下をご覧ください。

https://www.ss.cc.tohoku.ac.jp/n20240215-1/

（スーパーコンピューティング研究部，情報部デジタルサービス支援課）

令和 6 年度利用負担金について

令和 6 年度の利用負担金についてお知らせします。詳細は以下をご覧ください。

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

（情報部デジタルサービス支援課）

計算科学・計算機科学人材育成のためのスーパーコンピュータ無償提供制度について

東北大学サイバーサイエンスセンターでは、計算科学・計算機科学分野での教育貢献・人材育成を目的として、無料で大規模科学計算システムを利用できる制度を用意しております。提供の対象は、大学院・学部での講義実習等の教育目的(卒業論文、修士論文、博士論文での利用を除く)に限ります。利用を希望される場合は以下の情報を添えて、講義開始の 2 週間前までに edu-prog[at]cc.tohoku.ac.jp 宛お申し込みください。

・講義担当者氏名
・同所属
・同連絡先（住所，電話，電子メール）
・講義名
・講義実施日時（１セメスターの中で実習を予定している回数）
・センター端末機室等での実習利用希望の有無（必要であれば予定日時）
・講師派遣の希望の有無
・講義シラバス
・講義ウェブ（もし用意されていれば）
・受講者数（予定）
・必要とする理由（利用目的：例えば、数値シミュレーションの研修を行うなど）
・期待できる教育効果
・居住性チェックリストの提出（受講者に外国人が居る場合）
　参照：https://www.ss.cc.tohoku.ac.jp/apply-for-use/#toc3
・その他（センターへの要望等）

なお、講義終了後、報告書（広報誌 SENAC へ掲載）の提出をお願いいたします。

たくさんのお申し込みをお待ちしております。不明な点は、edu-prog[at]cc.tohoku.ac.jp までお問い合わせください。

（スーパーコンピューティング研究部，情報部デジタルサービス支援課）

大規模科学計算システムの機関（部局）単位での利用について

東北大学サイバーサイエンスセンターでは、大規模科学計算システムをご利用いただくにあたり、利用負担金を利用者単位のほか、機関（部局）単位で年間定額をお支払いいただくことで利用できるサービスも提供しております。このサービスは、機関（部局）単位でお申し込みいただくことにより、その構成員であれば、各研究室が個別に利用負担金を支払うことなく、下記システムを利用できる仕組みとなっております。

これまで計算機を利用する機会がなかった研究者による新たなニーズへの対応や研究室の計算機では実行できなかった大規模シミュレーションが実行可能であり、また自前で計算機を導入するためのコストや運用コストも削減可能です。すでにご利用いただいている機関（部局）からは、当初の予想を上回るご利用をいただき、ご好評をいただいております。

占有利用・共有利用については必要に応じて取り混ぜながら、ご予算に合わせて、年間定額により利用することが可能となっておりますので、ぜひご相談ください。

記

【利用可能なシステム】

・サブシステム AOBA-S
・サブシステム AOBA-A
・サブシステム AOBA-B
・ストレージシステム
・大判カラープリンター（光沢紙、ソフトクロス紙）

【問い合わせ先・申し込み先】
デジタルサービス支援課（cc-uketuke[at]grp.tohoku.ac.jp）

（情報部デジタルサービス支援課）

民間企業利用サービスについて

東北大学サイバーサイエンスセンターでは、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を、民間企業の方が無償または有償にてご利用頂ける制度を用意しております。本サービスにおける利用課題区分は以下の２つとなります。

・大規模計算利用(有償利用)
・トライアルユース(無償利用)

詳細については以下を参照ください。

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

【問い合わせ先・申し込み先】
デジタルサービス支援課（cc-uketuke[at]grp.tohoku.ac.jp）

（情報部デジタルサービス支援課）

— 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。

－用紙サイズ・文字サイズ等の目安－

・サイズ：A4
・余白：上＝30㎜　下＝25㎜　左右＝25㎜　綴じ代＝0
・標準の文字数（45 文字 47 行）
・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)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。詳細はデジタルサービス支援課までお問い合わせください。
(2)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。
(3)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。
(4)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　情報部デジタルサービス支援課
e-mail　cc-uketuke@grp.tohoku.ac.jp
TEL　　022-795-3406

スタッフ便り

今年１月に第一子となる娘が生まれ、家に子供がいる慌ただしい生活が始まりました。子供が成長するのは速いと以前より様々な方から伺ってはいましたが、娘の日々の細かい変化に驚かされています。昨日までは握りこぶししかできなかったのが今日は指をばらばらに動かせるようになっていたり、何だかぼんやりした表情だったのがいつの間にかはっきり笑いかけてくれるようになったり、どんどん人間らしくなっていく過程は大変興味深く、愛おしいです。あっという間に大きくなってしまうのが目に見えているので、今を大切に家族と暮らしたいと思います。（K. T）

今年から薬を飲み始めました。といっても、血圧を下げる薬でもなく、高値安定で時々限界を突破する尿酸値を下げる薬でもなく、花粉症の薬です。これまではスギ花粉によるアレルギー症状も特になく、今年の春は花粉が大量発生ですとニュースで聞いてもどこか他人事でした。たまに原因もわからず鼻水が止まらなくなることはありましたが、今年は寝ている時の鼻づまりもひどく、朝起きてからは鼻水が止まらず、目もかゆくなり、このような症状が続くため、何かおかしいぞということで薬を飲んだところ、だいぶ症状が緩和されました。桜が咲くころには花粉も終わりだと耳にしたことはありましたが、桜が散った今でも宮城の花粉はまだ多いとニュースサイトに出ています。来年から春がキライになりそうです。

さて、青葉山ではこの４月から 3GeV 高輝度放射光施設 NanoTerasu が稼働しました。実験装置から出力された計測データはスーパーコンピュータ AOBA のストレージに保存することが可能で、またスパコンでもデータ解析が可能です。ぜひ活用して頂ければと思います。（S. O）

【サイバーサイエンスセンター内　情報部情報デジタルサービス支援課スタッフ異動のお知らせ】
2024. 3. 31 付け
［転出］
早坂　和勝　会計係長（病院経理課資金経理係長へ）
2024. 4. 1 付け
［転入］
熊谷　和拓　課長補佐（監査室監査スタッフ（係長）から）
木村　優太　専門職員（岩手大学技術部情報技術部情報技術室システム開発グループから）

八木山ベニーランドにて（2024 年４月）

令和６年４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
PDF 作成　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-S	SX-Aurora TSUBASA Type 30A
サブシステム AOBA-A	SX-Aurora TSUBASA Type 20B
サブシステム AOBA-B	LX 406Rz-2

サーバとホスト名

フロントエンドサーバ（AOBA-S 用）	sfront.cc.tohoku.ac.jp
データ転送サーバ（AOBA-S 用）	sfile.cc.tohoku.ac.jp
ログインサーバ（AOBA-A,B 用）	login.cc.tohoku.ac.jp
フロントエンド（AOBA-A,B 用）	front.cc.tohoku.ac.jp
データ転送サーバ（AOBA-A,B 用）	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-S	連　続　運　転
サブシステム AOBA-A	連　続　運　転
サブシステム AOBA-B	連　続　運　転
各種サ ー バ	連　続　運　転
大判プリンタ	平日 9:00～21:00

サブシステム AOBA-S の利用形態と制限値

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	sxsf	1	1VE	1 時間/1 時間	96GB
共用	sxs	1～2048	8VE 単位で確保	72 時間/720 時間	96GB×VE 数
占有	個別設定				

※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-A の利用形態と制限値

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
共有	sx	1	1VE	72 時間/720 時間	48GB×VE 数
		2～256	8VE 単位で確保		
占有	個別設定				

※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

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			

※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.57 No.2 2024−4