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.58 No.2 2025−4

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)*	*見学希望の方はスーパーコンピューティングサポートユニット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	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］

有機超塩基が触媒する分子変換反応

— ホスファゼン塩基 *t*-BuP4 によるアニオン活性化効果の評価 —

笹本　大空[1]、是永　敏伸[2]、重野　真徳[1,3]
[1]東北大学大学院薬学研究科, [2]岩手大学理工学部化学・生命理工学科、[3]JST さきがけ

本研究ではホスファゼン塩基 *t*-Bu-P4 が示す特異なアニオン活性化効果について、密度汎関数理論(DFT)計算を用いて評価を行った。計算には Gaussian 16 (revision C.01)パッケージソフトを利用し、塩化合物における分子内相互作用の評価には NEDA(Natural Energy Decomposition Analysis)及び NCI(Non-Covalent Interaction)解析を用いた。

1. はじめに

ホスファゼンは窒素とリンが二重結合 (-N=P-) を形成した化合物の総称であり、有機化学において重要な塩基の 1 つとして利用される。その中でも、(-N=P-) 構造を 4 つ有するホスファゼン塩基 (*t*-Bu-P4) は、Brønsted 塩基性が極めて高く、その塩基性は有機リチウム反応剤に匹敵するほどである(**図 1**)。[1] さらに、*t*-Bu-P4 のプロトン化体([*t*-Bu-P4]-H$^+$)は、一般的な金属カチオンと比較して、非常に大きな分子サイズを有することから、弱配位性カチオンとして対アニオンの反応性を向上させる効果が期待される。著者の所属する研究室では、これらの特徴を活用して *t*-Bu-P4 を触媒とする様々な分子変換反応を開発しており、特に C–O 結合や C–F 結合などの化学的に安定な結合の変換反応に焦点を当てて研究に取り組んできた。例えば最近、*t*-Bu-P4 が多様な芳香族フッ素化合物の芳香族求核置換(S_NAr)反応を円滑に進行させる触媒として利用できることを報告した(**図 2**)。[2] 一般に S_NAr 反応では、アニオン性の求核剤の付加とフッ化物イオンの脱離という二段階の反応を経由することが知られており、適切な位置に電子求引基を有する芳香族化合物にのみ適用できると捉えられてきた。これに対して近年、付加と脱離が一段階で進行する協奏的な芳香族求核置換反応(CS_NAr)反応が見出されており、これまで反応性に乏しいとされていた電子豊富な芳香族化合物にも適応できることが示されている。[3] しかし、これまでの反応例は散発的であり、幅広い範囲の基質に対して有効な CS_NAr 反応の開発は行われていなかった。こうした背景のもと、著者らの研究グループでは、*t*-Bu-P4 が有する高いアニオン活性化効果を活用することで CS_NAr 反応を円滑に触媒することに成功した。

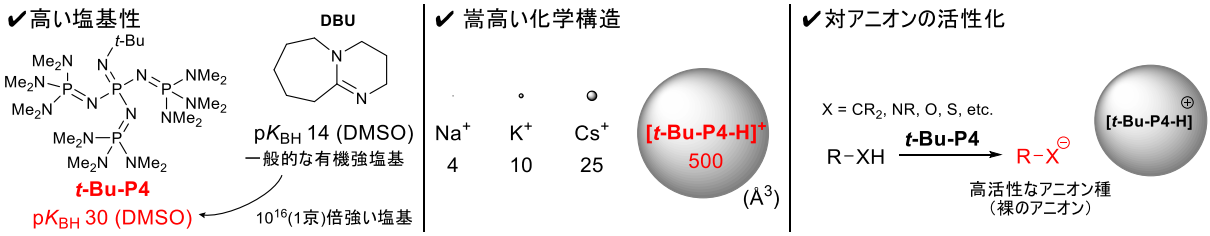


図 1　ホスファゼン塩基 *t*-Bu-P4 の特徴

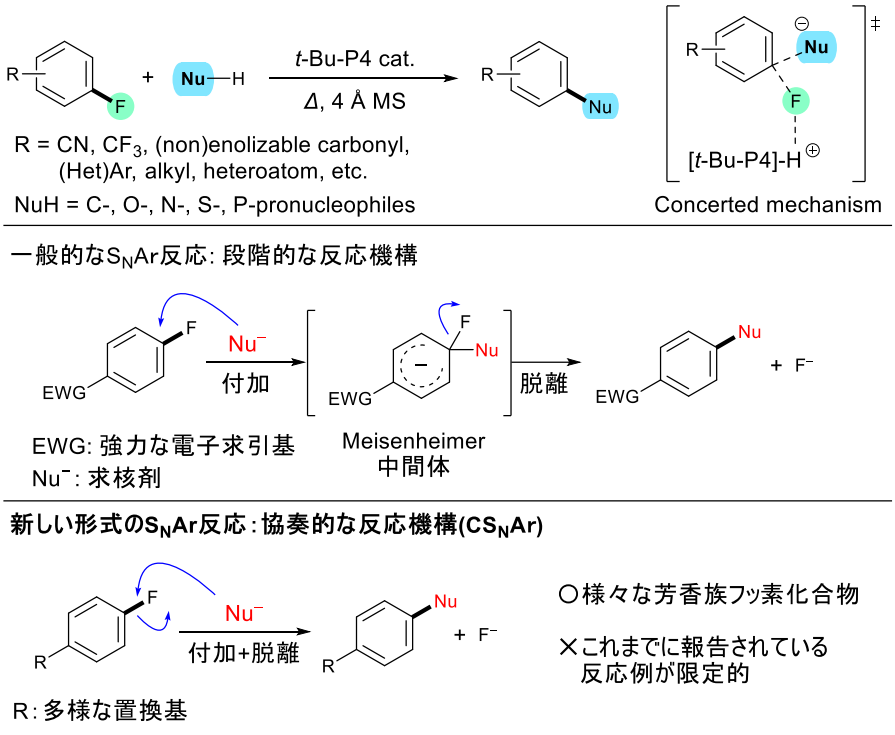


図 2　*t*-Bu-P4 触媒による芳香族求核置換(S_NAr)反応

2.　方法・結果：アニオン活性化効果の計算科学に基づく評価

本研究では *t*-Bu-P4 が有する特異なアニオン活性化効果を評価するために、東北大学サイバーサイエンスセンターの Gaussian 16 (revision C.01)パッケージソフトを利用して密度汎関数理論(DFT)計算を実施した。なお、NEDA(Natural Energy Decomposition Analysis)による相互作用エネルギーの評価には NBO 7.0 プログラムを使用した。

アニオンの反応性は対を成すカウンターカチオンとの静電相互作用の強さに依存する。すなわち、高い反応性を有するアニオンは、この相互作用が弱い状態(裸のアニオンともいう)にあるという事である。そこで、対カチオンに[*t*-Bu-P4]-H$^+$あるいは K^+/18-crown-6 を有する塩化合物 **1** 及び **2** について NEDA を行い、両化合物のカチオン-アニオン間に働く相互作用エネルギーの大きさを比較した(**図 3**)。化合物 **1** は **2** よりも相互作用エネルギーが小さく、高い反応性を示すアニオン種であることが示された。この傾向は、非共有結合相互作用(Non-Covalent Interaction、NCI)を可視化することでも確認でき、化合物 **1** においてカチオン-アニオン間の相互作用はより小さいことが示されている。さらに、化合物 **1** および **2** の HOMO(最高被占軌道)のエネルギー準位の比較からも、**1** の方が高い反応性を有していることが分かった。

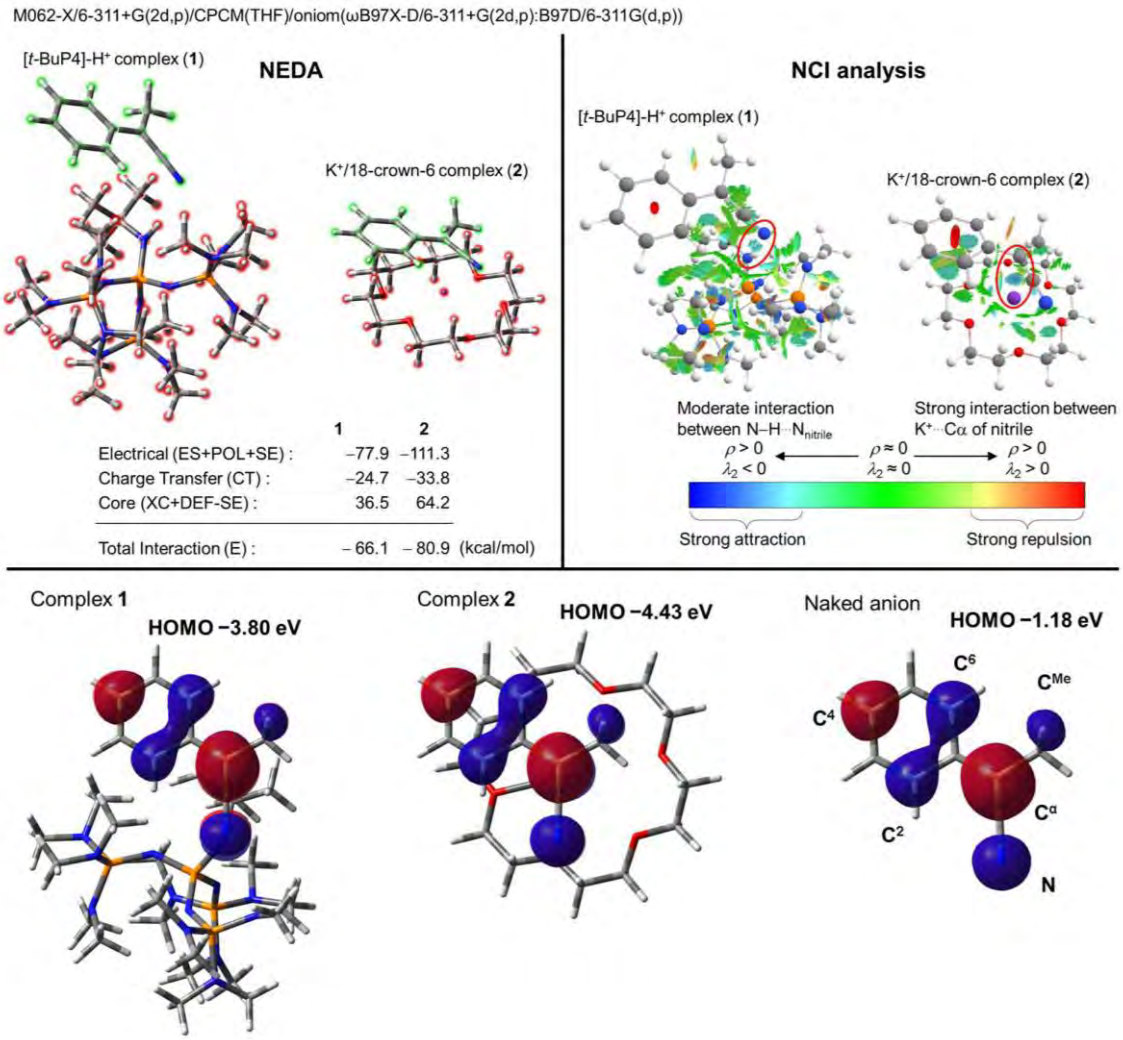


図 3　塩化合物 1 と 2 の相互作用の比較と HOMO のエネルギー準位の比較

本計算結果を実証するための実験として 4-フェニルフルオロベンゼンと 2-フェニルプロピオニトリルとの S_NAr 反応を実施したところ、*t*-Bu-P4 を触媒とする反応では目的の化合物を高い収率で与えた一方で、KO*t*-Bu/18-crown-6 を用いた反応では、ほとんど反応が進行しなかった(**図 4**)。この結果は化合物 **1** が高い反応性を示すという計算結果を支持している。

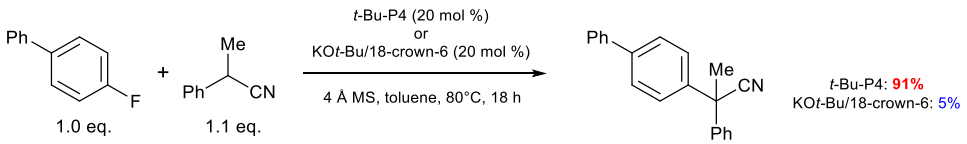


図 4　*t*-Bu-P4 と KO*t*-Bu/18-crown-6 の反応性の比較

3.　おわりに

本稿では DFT 計算に基づいて *t*-Bu-P4 が有するアニオン活性化効果の評価を実施し、化合物 **1** が高い反応性を有することが理論的に明らかとなった。今後、*t*-Bu-P4 を利用した反応系について、さらなる計算・解析を実施することで本触媒が示す特異な反応性の解明と新奇反応の創出が期待される。

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くと共に、令和 5 年度若手・女性研究者支援課題として利用負担金の一部をご支援いただいた。この場をお借りして御礼申し上げます。

参考文献

[1] R. Schwesinger, *et al*., Extremely Strong, Uncharged Auxiliary Bases; Monomeric and Polymer-Supported Polyaminophosphazenes (P2–P5), *Liebigs Ann*., 1055–1081, 1996.

[2] M. Shigeno, *et al*., Catalytic Concerted S_NAr Reactions of Fluoroarenes by an Organic Superbase, *J. Am. Chem. Soc*., 146, 32452–32462, 2024.

[3] S. Rohrbach, *et al.*, Concerted Nucleophilic Aromatic Substitution Reactions, *Angew. Chem. Int. Ed*., 58, 16368–16388, 2019.

［共同研究成果］

新規未分化 iPS 細胞抗体デザインを行うための分子動力学計算のパラメータ最適化

大野　詩歩（東北医科薬科大学 分子生体膜研究所）

抗体と糖鎖の相互作用を明らかにするための一般的な手法として X 線結晶構造解析があるが、糖鎖は結晶化が困難であることなどの問題点がある。そこで本研究では、未分化 iPS 細胞抗体-糖鎖複合体モデルの構築および相互作用解析による抗体の糖鎖認識機構の解明を目的とし、分子動力学計算を実施した。

1.　序論

抗体のリガンド結合様式を原子レベルで解析する手法として、X 線結晶構造解析が広く用いられている。しかし、糖鎖は電子密度が低く、像を得ることが困難な傾向にある。近年、計算化学の分野は急速に発展しており、その代表例として AI を用いたタンパク質モデリングシステムである AlphaFold が挙げられる。しかし、AlphaFold では、学習データとなる修飾糖鎖の結晶構造が限られなど、未だ発展途上であり、修飾糖鎖を含むタンパク質の構造予測精度は低い。また、抗体や糖鎖の動的挙動を調べるにはモデル構築だけでなく、分子動力学（MD）計算が不可欠である。本研究では、難易度の高い抗体-リガンド糖鎖複合体のモデルを構築し、その結合様式を計算化学と NMR を組合わせ、解明した。

2.　方法

抗体-糖鎖複合体の構築を行うため、リガンド糖鎖として lacto-*N*-fucopenaose Ⅰ（LNFP Ⅰ）(Fucα(1-2)Galβ(1-3)GlcNAcβ(1-3)Galβ(1-4)Glc)、抗体は R-17F および R-13E を用いた[1]。ドッキングモデルの構築にあたり、PDB に登録されている LNFP Ⅰの結晶構造を抽出し、グリコシド結合まわりの二面角の傾向を抽出し、安定なコンホメーションを調査した。R-17F および R-13E 抗体の Fv 領域の立体構造モデルは、AlphaFold により作成した[2]。作成した抗体は、構造最適化のため、MD シミュレーションを行った。MD シミュレーションは、力場に CHARMM-GUI を用い[3]、計算ソフトには GENESIS 1.7.0[4, 5]を用いて、100 ns 実施した。複合体のエピトープを実験的に同定するため、化学合成した LNFP Ⅰと R-17F あるいは R-13E の混合溶液に対して、STD-NMR を測定した。LNFP Ⅰの NMR 信号の帰属には 1D-^{1}H、2D DQF-COSY、CLIP-COSY、HOHAHA、NOESY、1D-^{13}C、^{1}H-^{13}C HSQC、^{1}H-^{13}C HMBC を用いた。これらの結果を踏まえ、LNFP Ⅰと R-17F および R-13E 抗体とのドッキングシミュレーションを行った。ドッキングシミュレーションは、HADDOCK2.4 を用い STD-NMR とデータベース解析の結果を組み込んで行った[6, 7]。得られたドッキングポーズのクラスタリングを行い、妥当と思われる複合体構造の最適化のために MD シミュレーションを実施した。

3.　結果・考察

PDB に登録されている LNFP Ⅰのグリコシド結合まわりの各二面角を求めたところ、その分布は非常に限局されていたことから、LNFP Ⅰの構造は柔軟性が乏しいことが推察された(図 1)。

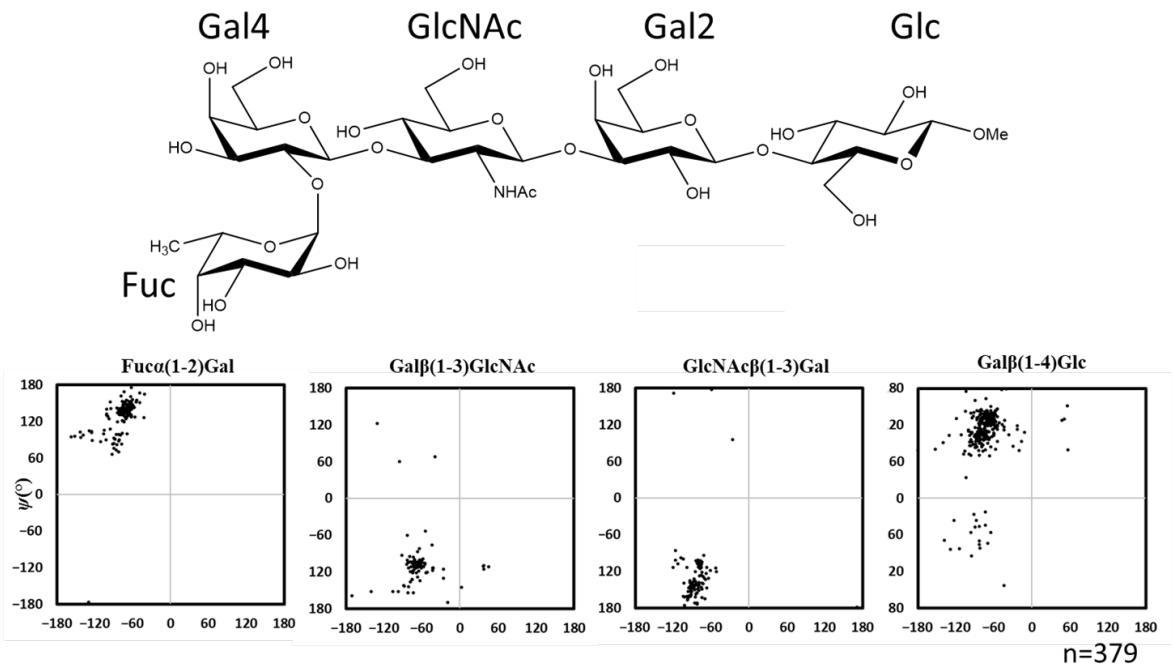


図 1. PDB に登録されている LNFP Ⅰの二面角の分布。六員環の単糖（i）と単糖（i−1）が 1-4 結合でつながっている場合、φ (O5-C1(i)-O4(i)-C4(i-1))、ψ (C1(i)-O4(i)-C4(i-1)-C3(i-1))と定義した。

2 次元 NMR により LNFP Ⅰの NMR シグナルの帰属を行い、各抗体と LNFP Ⅰの STD-NMR の解析を行った。その結果、R-17F および R-13E はいずれも LNFP Ⅰの非還元末端側を共通のエピトープとして認識していることが示唆された。また、STD-NMR のシグナルは R-13E の方が R-17F よりも強かったため、R-13E と LNFP Ⅰの結合・解離は R-17F の場合と比べて速いことが示唆された（図 2)。

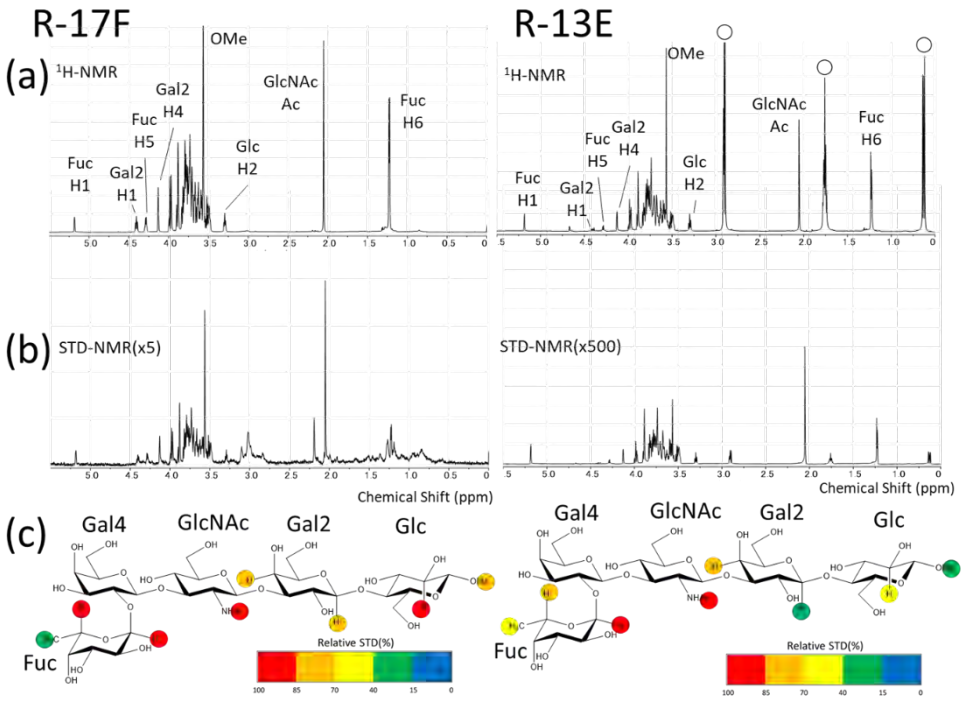


図 2. R-17F または R-13E と LNFP Ⅰの STD-NMR 解析。〇は DSS シグナル。(a)リガンド糖鎖のみで測定したスペクトル。(b)STD-NMR 測定で得た差スペクトル(c)各プロトンの強度比。強度比は(差スペクトルの強度)/(コントロールスペクトルの強度）として算出した。強度比が一番大きい GlcNAc の Ac プロトンを 100%とし、色分けを行った。STD-NMR のシグナル強度は R-17F においてはコントロールの 5 倍、R-13E においてはコントロールの 500 倍にした。

LNFP I-抗体複合体モデルを作成するため、ドッキングシミュレーションを行った。さらに得られた複合体の最適化のため、100 ns の MD シミュレーションを行った。その結果、R-17F-LNFP I 複合体は結合を維持していたが、R-13E-LNFP I 複合体は 100 ns のシミュレーション時間の間、結合を維持することができなかった(図 3)。

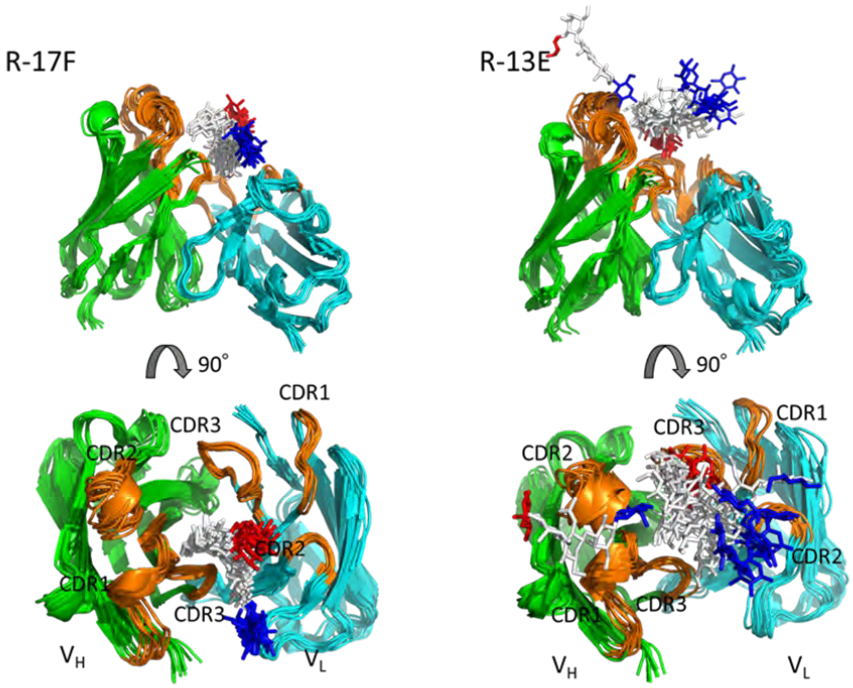


図 3. MD シミュレーション(100 ns)後の複合体の構造。各スナップショットは 10 ns ごとに抽出した構造である。緑は H chain、水色は L chain、橙色は CDR、赤は Fuc、青は Glc を示す。

H chain と L chain のどちらが結合安定性に寄与しているかを評価するため、(1)H chain は R-17F の配列であり、Lchain は R-13E の配列を有するモデルと、(2) Hchain は R-13E の配列であり、Lchain は R-17F の配列を有するモデルの 2 つに対し MD シミュレーションを行ったところ、(2)のモデルのみ結合を維持していた。したがって結合には R-17F の L chain のアミノ酸残基がリガンドとの結合に重要であることを示している。

4.　結論

本研究は、困難とされる糖鎖-抗体複合体の構築と、その結合様式の解明を計算化学と NMR を組み合わせて明らかとした。この手法は他の組み合わせの抗体-糖鎖複合体に対しても応用でき、新規抗体デザインに貢献することができる。

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活用することで実現することができた。また、研究遂行にあたり、同センターの関係各位より貴重なご指導とご協力を賜ったことに深く感謝申し上げる。

さらに、MD シミュレーションにご協力いただいた Re Suyong 博士（NIBIOHN）、ならびに NMR 測定に多大なご支援をいただいた 松木智之様、佐藤慎一様（東北医科薬科大学） に対し、心より感謝の意を表する。

参考文献

[1] H. Nakao, et al., Characterization of novel antibodies that recognize sialylated keratan sulfate and lacto-*N*-fucopentaose I on human induced pluripotent cells: comparison with existing antibodies, *Glycobiology*, **33**, 150-164 (2023)

[2] J.Jumper, R.Evans, A.Pritzel, et al. Highly accurate protein structure prediction with AlphaFold, *Nature*, **596**, 583–589 (2021)

[3] F.A. Momany and R.Rone, Validation of the general purpose QUANTA ®3.2/CHARMm® force field, *J. Comput. Chem.*, **13**, 888–900 (1992)

[4] C. Kobayashi, J. Jung, Y. Matsunaga, T. Mori, T. Ando, K. Tamura, M. Kamiya, and Y. Sugita, GENESIS 1.1: A hybrid-parallel molecular dynamics simulator with enhanced sampling algorithms on multiple computational platforms, *J. Compute. Chem.*, **38**, 2193-2206 (2017)

[5] J. Jung, T. Mori, C. Kobayashi, Y. Matsunaga, T. Yoda, M. Feig, and Y. Sugita, GENESIS: a hybrid-parallel and multi-scale molecular dynamics simulator with enhanced sampling algorithms for biomolecular and cellular simulations, *Wiley Interdiscip Rev. Comput. Mol. Sci.*, **5**, 310-323 (2015)

[6] C. Dominguez, R. Boelens, and Alexandre M. J. J. Bonvin, HADDOCK: A Protein−Protein Docking Approach Based on Biochemical or Biophysical Information, *J. Ame. Chem. Soc.*, **125**, 1731-1737 (2003)

[7] S.J. de Vries, A.D.J. van Dijk, M. Krzeminski, M. van Dijk, A. Thureau, V. Hsu, T. Wassenaar and A.M.J.J. Bonvin, HADDOCK versus HADDOCK: New features and performance of HADDOCK2.0 on the CAPRI targets, *Proteins.*, **69**, 726-733 (2007)

［共同研究成果］

マイクロバブル径によるジャーナル軸受の摩擦特性と気液二相流解析

山崎佑人，川本裕樹，落合成行，畔津昭彦
東海大学

環境問題の解決に向けて，自動車は燃費の向上が要求され，本研究の対象であるジャーナル軸受も摩擦の低減が必要である．そこで新たな摩擦低減手法としてマイクロバブルに着目しており，マイクロバブル径がジャーナル軸受の摩擦特性に与える影響を気液二相流解析から調査した．軸受すきまより大きいバブル径においても新たに摩擦低減効果を確認でき，バブル径の増加により，低減率は向上する結果となった．

1.　緒言

近年，地球温暖化のような環境問題やエネルギー問題は社会的な課題とされている．そのため，自動車では燃費の向上が強く求められ，燃費の向上にはエネルギー損失の低減が重要である．また，エンジンの摩擦損失は自動車全体のエネルギー損失の約 2 割[1]を占めるとされている．特にエンジンは多くの部品で構成されることから摺動部が多く，本研究の対象であるジャーナル軸受を含む各機械要素で摩擦低減が必要となる．ここでジャーナル軸受とは，回転軸と軸受のすきまに供給される潤滑油から生じる油膜圧力により，軸を非接触で支持する機械要素である．特徴として負荷容量が高く，耐衝撃性や耐摩耗性に優れていることから，自動車だけでなく高速回転機械にも多く利用されている．摩擦低減手法として，従来は潤滑油の低粘度化[1]や添加剤の使用がされてきたが，これらは焼付きや環境に悪影響を及ぼす可能性が考えられる．そこで環境に低負荷な摩擦低減手法が求められる．

本研究は新たな手法としてマイクロバブルに着目している．マイクロバブルは直径数 μm から 100 μm 程度の微細な気泡を指す[2]．海洋分野では船底にマイクロバブルを発生させることで摩擦抵抗の低減が確認されている[3]．また，洗浄や医療分野など多岐にわたる分野[2]でマイクロバブルの応用に向けた研究が行われている．

そこで本研究はジャーナル軸受の潤滑油にマイクロバブルを応用する取り組みを行っている．これまでの成果から，マイクロバブルの混入による摩擦の低減が確認されており，同心状態では軸受すきま（以下，すきまと称する）内のバブルの挙動が実験と二相流解析で一致する傾向を得られている[4, 5]．さらに二相流解析の結果から，マイクロバブルの混入で壁面近傍の速度勾配が緩和し，摩擦が低減したと考えられている[5]．しかし，摩擦低減メカニズムについては未だ明らかになっていない点も多くある．よって，本研究はマイクロバブルによる摩擦低減メカニズムの解明を目的とし，本稿ではマイクロバブル径に着目した．これまではすきまより小さいバブル径を対象としていた．しかし，摩擦低減率の向上も新たに本研究の目的とし，すきまより大きいバブル径も含めたジャーナル軸受の摩擦特性に与える影響を二相流解析から調査した．

2.　計算手法

本研究は支配方程式として式(1)に示す連続の式および式(2)に示す非圧縮性 Navier-Stokes 方程式を用いた．

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

$$\frac{\partial \mathbf{U}}{\partial t} + \{(\mathbf{U} - \mathbf{U}_0) \cdot \nabla\}\mathbf{U} = -\frac{1}{\rho}\nabla P + \frac{\mu}{\rho}\nabla^2 \mathbf{U} \tag{2}$$

ここで$\mathbf{U}$は流体の速度，$\mathbf{U}_0$は格子の移動速度，ρは密度，Pは圧力，μは粘性係数を示す．本研究は

単一のバブルの挙動および周辺の流れ場に着目する．このため移動するバブルに計算領域が追従するよう ALE（Arbitrary Eulerian and Lagrangian）法に基づき，式(2)に格子の移動速度$\mathbf{U}_0$を考慮した．なお$\mathbf{U}_0$は気相領域の平均速度とした．また速度と圧力の連成は Fractional step 法を用い，空間の離散化は等間隔直交格子上で行う．空間の離散化手法として式(2)に示す移流項は数値振動の抑制に利点を有する高次精度スキームである 5 次精度 WENO（Weighted Essentially Non-Oscillatory）法，粘性項には 2 次精度中心差分法を用いた．また，時間の離散化手法は移流項および粘性項ともに 3 次精度 TVD Runge-Kutta 法を用いた．圧力の Poisson 方程式は SOR 法で解いた．

気液界面と物体形状は Level set 法[6, 7]で定義しており，気液界面の時間変化は式(3)に示す移流方程式から求められる．

$$\frac{\partial \phi}{\partial t}(\mathbf{U}-\mathbf{U}_0)\cdot\nabla\phi = 0 \tag{3}$$

ϕは気液界面に対する Level set 関数を指し，空間の離散化は 5 次精度 WENO 法，時間の離散化は 3 次精度 TVD Runge-Kutta 法を用いた．しかし，式(3)の移流に伴い，Level set 関数は適切な距離関数としての性質の維持が困難となるため，界面近傍を除く計算格子は式(4)を解くことで Level set 関数の再初期化を行う．なお，τは Level set 関数の更新に関する疑似時間の変数を指す．

$$\frac{\partial \phi}{\partial \tau} = \mathrm{sign}(\phi)(1-|\nabla\phi|) \tag{4}$$

また，界面付近の不連続な物理量に対応するため，境界近傍の計算格子に任意の境界条件を与える Ghost fluid 法[6, 7]を用いた．本研究においては液相の速度，気相の法線方向速度および圧力は外挿して求め，これらの値を示すq_{Gf}は式(5)を解くことで決定される．表面張力は液相の圧力境界条件として Laplace 圧力の関係式を考慮し，式(6)よりジャンプ条件として与えられる．

$$\frac{\partial q_{\mathrm{Gf}}}{\partial \tau} + \mathrm{sign}(\phi)\frac{\nabla\phi}{|\nabla\phi|}\cdot\nabla q_{\mathrm{Gf}} = 0 \tag{5}$$

$$P_{\mathrm{Gf,l}} = P_{\mathrm{Rf,g}} + \sigma\kappa \tag{6}$$

ここでσは表面張力，κは界面の曲率を示す．

本研究は移動する物体形状を Level set 関数で定義したが，物体表面の Non-Slip 条件の決定には Ghost cell 法[6, 7]を用いた．本手法は物体内部の境界に設けられた計算格子を Ghost cell として，Ghost cell の値の決定は Image point を用いる．Ghost cell の速度$\mathbf{U}_{\mathrm{Gc}}$は式(7)から求められる．

$$\mathbf{U}_{\mathrm{Gc}} = \mathbf{U}_{\mathrm{Ip}} - \frac{d_{\mathrm{Ip}}+d_{\mathrm{Gc}}}{d_{\mathrm{Ip}}}\left(\mathbf{U}_{\mathrm{Ip}}-\mathbf{U}_{\mathrm{Ib}}\right) \tag{7}$$

ここで Image point の速度$\mathbf{U}_{\mathrm{Ip}}$はその点を囲む 8 つの格子点から Tri-linear 内挿によって算出される．d_{Ip}はプローブの長さを示し，本研究は$1.75\Delta x$とした．d_{Gc}は Ghost cell から物体表面までの法線距離，$\mathbf{U}_{\mathrm{Ib}}$は物体の移動速度である．圧力と Level set 関数は Image point の値を Ghost cell に用いることで，物体近傍で Neumann 条件を満たすように決定した．

3. 計算条件

本解析は軸受全体を模擬するのではなく，ある微小領域を対象とした三次元解析を実施した[8]．図 1 は計算領域であり，50 μm のバブル径を例に示す．本来，軸と軸受は円筒形状であり曲率を

有するが，微小領域において曲率の影響は小さいと考え，考慮していない．計算領域の上下には各 5 格子の物体を設け，境界条件として上側の壁面は軸の回転を模擬するため周速度を付与した．また，下側の壁面は軸受を想定していることから静止状態とした．その他の境界は周期境界条件を用いている．計算条件は表 1 に示し，流体の物性は空気および粘度グレード ISO VG32 相当のオイルを想定して設定した．また軸の直径や回転数は実験[4]での条件に基づき決定した．

本解析は同心状態を想定し，すきま内に単一のバブルを配置した．すきまより小さいバブル径である 30 μm と 40 μm の 2 ケースは初期形状を球体で定義した．しかし，すきま以上のバブル径である 50 μm から 70 μm の 3 ケースは初期形状を球体で定義することができない．そのため，球体と同一の体積となる楕円体で定義をした．本研究では壁面せん断応力の平均値を用いて摩擦力の評価を行う．この基準を明確化するため，流体領域に占めるバブルの体積割合（ボイド率）が 2.0 % となるように計算領域を設定した．よって，計算領域の格子点数はバブル径で異なり，最小である 30 μm は 1,946,637 点，最大となる 70 μm で 24,667,412 点である．なお，格子幅は格子依存性の検証から 0.75 μm と決定した．本解析は MPI を用いた並列計算により実施し，32 並列とした．

表 1　計算条件

		Air	Oil
Density	[kg/m^3]	1.2	868.5
Viscosity	[Pa･s]	1.8×10^{-5}	2.805×10^{-2}
Surface tension	[mN/m]	33.3	
Shaft diameter	[mm]	50	
Rotational speed (Peripheral velocity	[rpm] [m/s])	2000 (5.24)	
Bearing clearance	[μm]	50	
Bubble diameter	[μm]	30, 40, 50, 60, 70	
Void fraction	[%]	2.0	
Grid size	[μm]	0.75	
CFL Number		0.3	

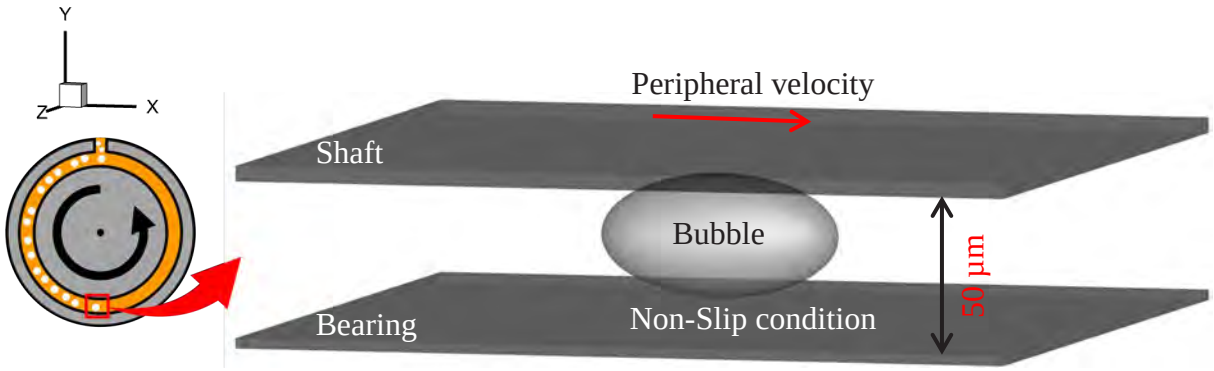
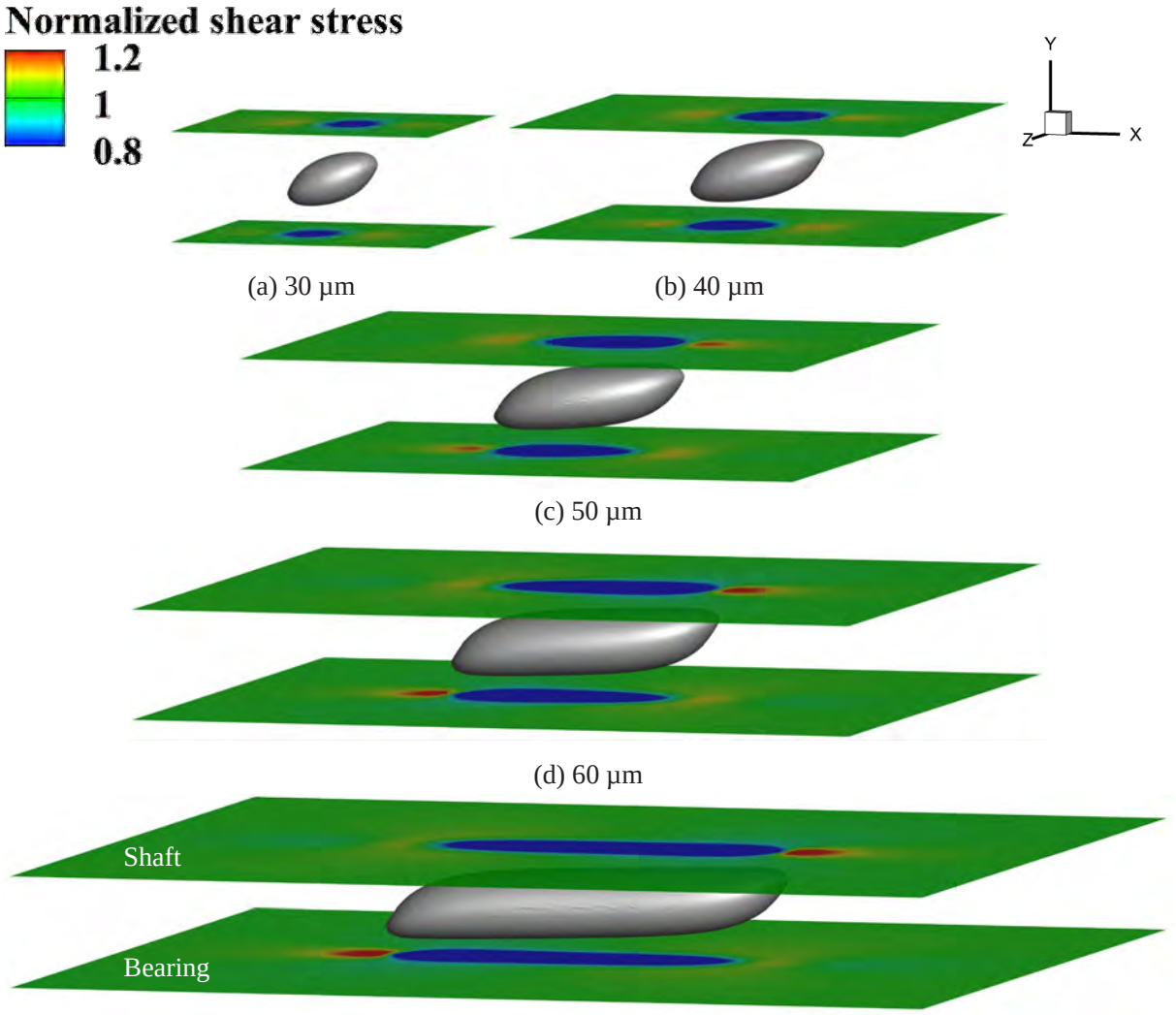


図 1　計算領域

4.　結果および考察

図 2 に無次元せん断応力で可視化した結果を示す．せん断応力の無次元化は単相流（バブルなし）のせん断応力を用いて実施した．可視化結果より，全ての条件でバブルは流れ方向に延伸する挙動を示しており，バブル径の増加に伴い，変形は大きくなっていることが分かる．またせん断応力に着目すると，変形したバブルの直上で低減効果を確認することができた．一方，バブルの先端ではせん断応力の増加も生じている．図 3 に各バブル径における壁面せん断応力の平均値から低減率を算出した結果を示す．バブル径の増加に伴い，低減率が向上する結果を得られており，最大で 2.38 %の低減効果を確認できた．傾向はバブル径の増加に対して，低減率の上昇割合が徐々に小さくなることから，あるバブル径で低減率は収束すると推測される．



(a) 30 μm (b) 40 μm

(c) 50 μm

(d) 60 μm

(e) 70 μm

図 2　各バブル径における無次元せん断応力分布

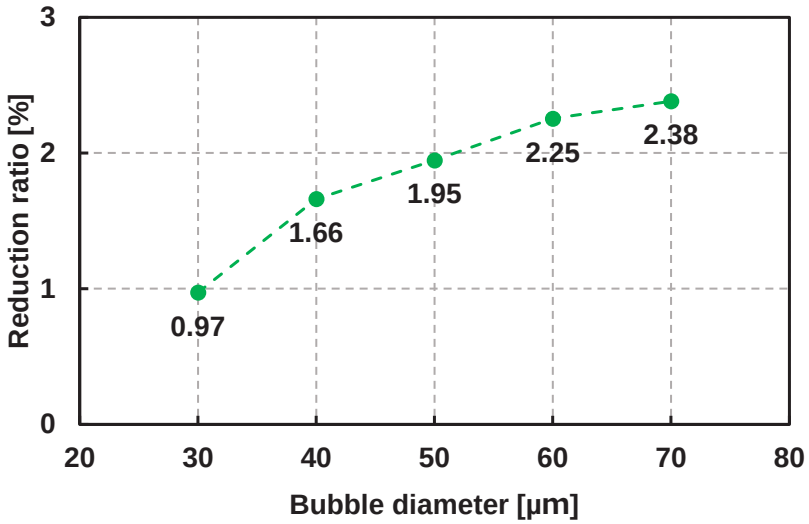
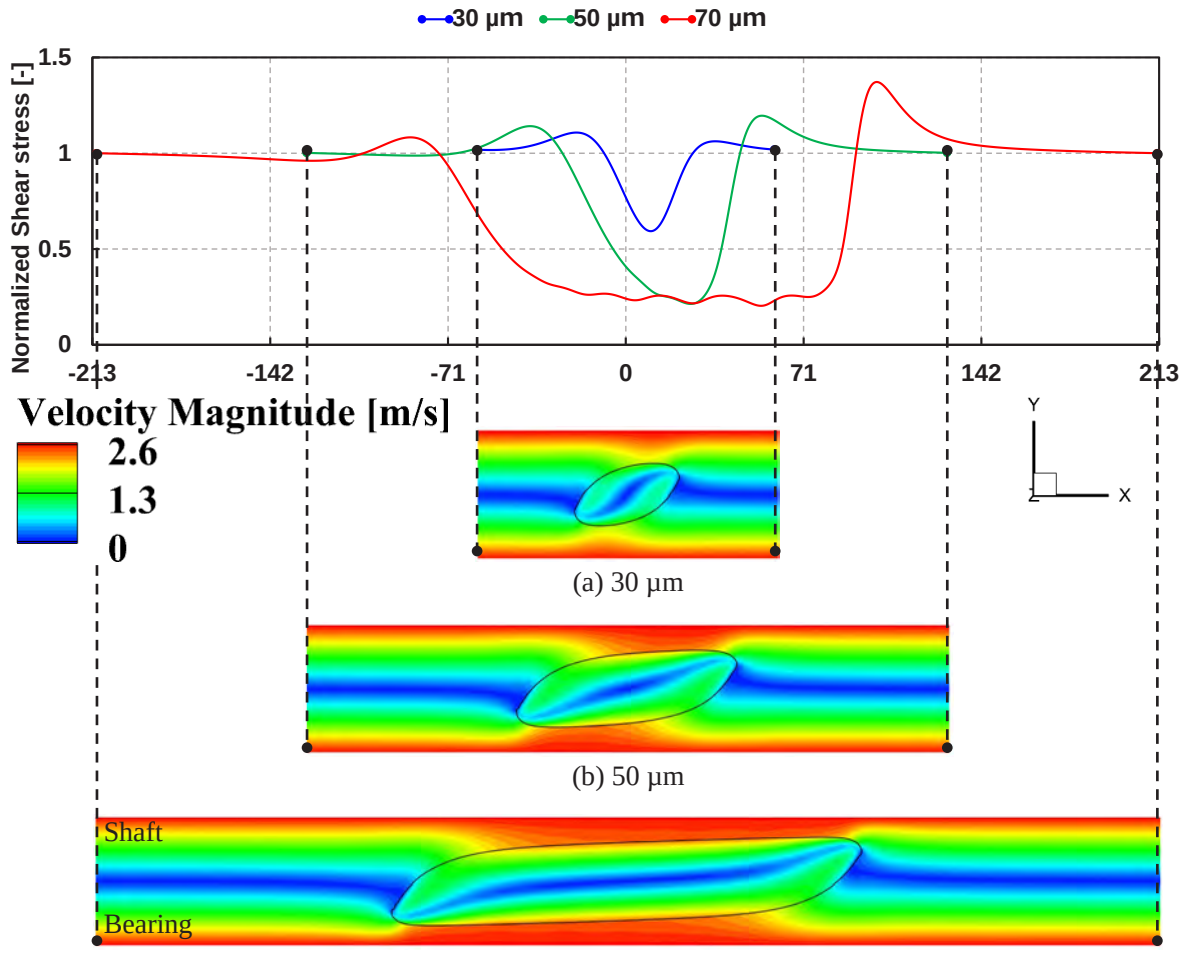


図 3　バブル径と低減率の関係

図 4 に速度の絶対値で可視化した結果とその直上における無次元せん断応力の 1 次元分布（X 方向）を示す．なお，速度の絶対値は Z 方向の計算領域に対して中心で可視化している．ここではすきまより小さい 30 μm，すきまと同等の 50 μm，すきまより大きい 70 μm の計 3 ケースの結果を示し，すきま以下とすきま以上のバブル径に分けて説明する．まず，すきま以下のバブル径である 30 μm と 50 μm に着目すると，局所的な最大低減率が異なり，バブル径の増加で最大低減率も向上している．可視化図から，30 μm と 50 μm ではバブルの界面と壁面までの距離（Y 方向）が異なり，バブル径の増加で狭くなり，壁面との速度差も小さいことが分かる．よって，すきま以下のバブル径においてはバブル径の増加に伴い，局所的な最大低減率が上昇し，低減率の向上に繋がったと考えられる．次にすきま以上のバブル径である 50 μm と 70 μm においては前者と異なり，局所的な最大低減率は変化していない．また，可視化図からもバブル径による，界面と壁面までの距離に差が生じていないことを定性的に確認できる．特にすきま以上のバブルでは，すきまの大きさによる制約を受け，変化しなかったと考えられる．一方，X 方向に対する変形は顕著に確認でき，バブル径が増加することで低減領域は拡大し，低減率の向上に寄与したと考える．



(a) 30 μm

(b) 50 μm

(c) 70 μm

図 4　速度の絶対値分布および直上の無次元せん断応力分布

5.　結言

本研究はマイクロバブルによる摩擦低減メカニズムの解明および摩擦低減率の向上を目的に，すきま以上の大きいバブルも含めたバブル径が摩擦特性に与える影響を気液二相流解析から調査した．全てのバブル径でボイド率を統一したことで，バブル径のみが与える影響を評価でき，バ

ブル径の増加に伴う，摩擦低減率の向上を確認した．本解析では単一のバブルとしたが，今後は回転数の変更や複数のバブルを考慮したバブル同士の相互的な影響を調査する必要がある．

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得られた結果であり，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．また，本研究は自動車用内燃機関技術研究組合（AICE）の委託事業として得られた成果である．

参考文献

[1] 中村隆，トライボロジー技術の進展による自動車の省エネ，トライボロジスト，第 61 巻，第 2 号，65-70，2016

[2] 芹澤昭示，マイクロ／ナノバブルの基礎，日本マリンエンジニアリング学会誌，第 46 巻，第 6 号，56-61，2011

[3] T. Takahashi, A. Kakugawa, Y. Kodama and M. Makino, Experimental Study on Drag Reduction by Microbubbles Using a 50m-Long Flat Plate Ship, 2nd International Symposium on Turbulence and Shear Flow Phenomena, 175-180, 2001

[4] 小谷晋平，田代祐樹，川本裕樹，高橋俊，落合成行，ジャーナル軸受の摩擦特性に及ぼす潤滑油中のマイクロバブルの影響 ―第 1 報　実験的検討―，トライボロジー会議 2020 秋別府予稿集，528-531，2020

[5] 川本裕樹，小谷晋平，アプサラセイタン，高橋俊，落合成行，ジャーナル軸受の摩擦特性に及ぼす潤滑油中のマイクロバブルの影響 ―第 2 報　二相流 CFD 解析と摩擦低減メカニズムの考察―，トライボロジー会議 2020 秋別府予稿集，532-533，2020

[6] Y. Kawamoto, S. Takahashi, M. Ochiai, A. Azetsu and K. Yamamoto, Prediction of Oil Behavior in Piston Ring Groove Based on Gas-Liquid Two-Phase Flow Analysis, Journal of Advanced Mechanical Design, Systems, and Manufacturing, 14, 6, 2020

[7] S. Takahashi, T. Nonomura and K. Fukuda, A Numerical Scheme Based on an Immersed Boundary Method for Compressible Turbulent Flows with Shocks: Application to Two-Dimensional Flows around Cylinders, Journal of Applied Mathematics, 2014

[8] 山崎佑人，川本裕樹，吉村友輝，加々美昌樹，山崎浩作，畔津昭彦，落合成行，ジャーナル軸受すきま内におけるマイクロバブル径と摩擦特性に関する研究（第 2 報　気液二相流解析を用いたマイクロバブル径が与える影響），トライボロジー会議 2024 春東京予稿集, 3-4, 2024

［共同研究成果］

仮想光子場揺動モデルによる格子ガス法流体解析の可能性

――― 仮想光子のやりとり頻度で流体粘性を制御

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。すでに２回、SENAC 誌において研究の途中成果を報告[1,2]している。今回は、これらの研究成果を総合してさらに改良した“格子ガス法流体解析モデル”について述べる。

今回検討したモデルでは、まず“仮想光子場”を導入する。これは、非平衡統計力学において“揺動散逸定理”を導く際に仮定される“揺動場”という物理モデルをヒントにしている。“仮想光子場”は、流体が存在する空間のすべての場所に“背景”のように存在していて、流体を構成する“仮想粒子”との間で“運動量”と“エネルギー”をやりとりし流体を“揺動”させる。このやりとりのタイミングをうまく制御することによって、仮想粒子集団の内部で生じる“連行離脱挙動”の発生頻度を自由に変えて粘性の発現を制御する方法を提案する。

速く流れる粒子集団は、近傍の遅い粒子を加速したり、逆向きに動く粒子を静止させたりという形で“連行”の効果を周辺に及ぼす。また、その速さに付いていけなかった粒子を “離脱”する場合もある。このとき必要な“運動量”と“エネルギー”のやりとりは、背景にある“仮想光子場”との間で行う。この“連行離脱挙動”は、分子レベルのミクロスケールでは「分子間に働く“分子間力”の総合的な効果の反映」であり、流体を連続体として見るマクロスケールでは「“粘性特性”の発現の原因」と考えられる。格子ガス法の仮想粒子モデルは、仮想的なものではあるが、スケール的には、これらの中間の“メゾスコピックなスケール”に位置付けられる。従って、メゾスコピックスケールで流体挙動を特徴づけるパラメータとして“連行離脱確率”に注目し、そのタイミングを調整することによって、マクロスケールで発現する流体粘性を自在に制御することが期待できる。

“連行離脱挙動”のタイミングを調整する方法はいろいろなものを考案できる。本稿では、ひとつの極力簡易な計算アルゴリズムを提案する。また、これを「四角柱の遠方後流までのシミュレーション」に適用した事例を検討し、その応用可能性を考察する。

１．仮想光子場揺動モデル

（１）“仮想光子場”のイメージ

“仮想光子場”については、筆者の報告[2]において既に紹介している。そこでの記述と多少重複するが、以下のような“場”の存在を仮定する。

“仮想光子”が、流体の存在する空間中の全ての格子点の背景において、ある一定数存在していて互いに衝突散乱を繰り返していると考える。“仮想光子”は、“運動量”と“エネルギー”をもつが、“静止質量”はもたない。これらの“仮想光子”は、流体を模擬している通常の“仮想粒子”との間で相互作用を起こすことで“粘性”が関与する流体挙動をいろいろな強度で発現させる。

ここでは、モデルを極力簡易化するため、２つの過程のみを考える。ひとつは、「①“仮想光子”と“静止仮想粒子”が合体して“運動仮想粒子”を生む過程」である。もうひとつは、その逆の場合であり、すなわち、「②“運動仮想粒子”が“静止仮想粒子”と“仮想光子”に分離する過程」である。これを**図１**に示す。

“仮想光子場”のイメージは、まさに“通常の光が満ちた空間”のようであり、目に見えない背景空間をなしていると考える。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とエネルギーをもらう。運動している仮想粒子を静止させたい時は、“運動仮想粒子”から“仮想光子”を分離して、背景空間に運動量とエネルギーを返却する。ただし、このときの運動量とエネルギーのやりとりは、“仮想光子場”の“揺動”の結果生じたものと考える。“揺動”とは、平均値を中心にしたプラスマイナスの変動であり、ある時空間の範囲で合計するとゼロになる変動である。従って、流体が存在する領域においては、「“静止仮想粒子”と“仮想光子”の合体過程とその分離過程は、疎視化を行う時空間のスケールにおいて、互いに平衡して同じ回数だけ生じている。」という前提になる。

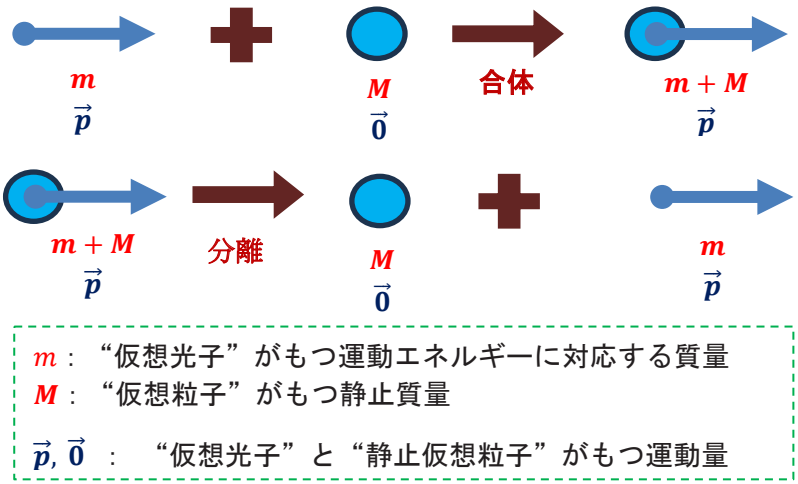


図１．“静止仮想粒子”と“仮想光子”の合体分離過程

（２）非平衡統計力学からのヒント

非平衡統計力学における流体モデル[3]では、次のような解釈がなされている。

「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。・・・あるときに揺動場を大きくするようなプラスの揺動力が作用したとすれば、平均的には、次には揺動場を小さくするようなマイナスの揺動力が作用してしまい、結果としては効果はゼロという意味である。しかし、たまたま同じ時刻に同じ場所で同じ種類の揺動力が２回同時に作用する場合もありうる。このような場合には揺動力は効果をもつ。」（cf. P105[3]）

そこで、“仮想光子場揺動モデル”では、仮想粒子集団の運動にある向きの揺動が生じたとき、それを検知して、続けて２回目の揺動を同じ向きに与えることにする。このときの２回目の揺動を、仮想光子との相互作用によって意図的にある確率的で生じさせる。この発想は、上述のとおり、非平衡統計力学における流体モデルと整合している。

２．“連行離脱過程”の追加とそれを実現する計算アルゴリズム

（１） 全体の計算手順

一般に、格子ガス法における基本的な計算過程は、次の２過程の繰り返しである。

① “仮想粒子”がその速度に従って格子点間を移動する“並進移動過程”

② いろいろな向きから格子点に到着した“仮想粒子”が、他の向きからやってきた“運動仮想粒子”やその格子点に存在していた“静止仮想粒子”と衝突散乱して速度分布の異方性が緩和され、その新しい速度分布に従って、それぞれの向きに出発する“衝突散乱過程”

ここでは、**図２**に示すように“衝突散乱過程”の直後に、“連行離脱過程”を追加する。

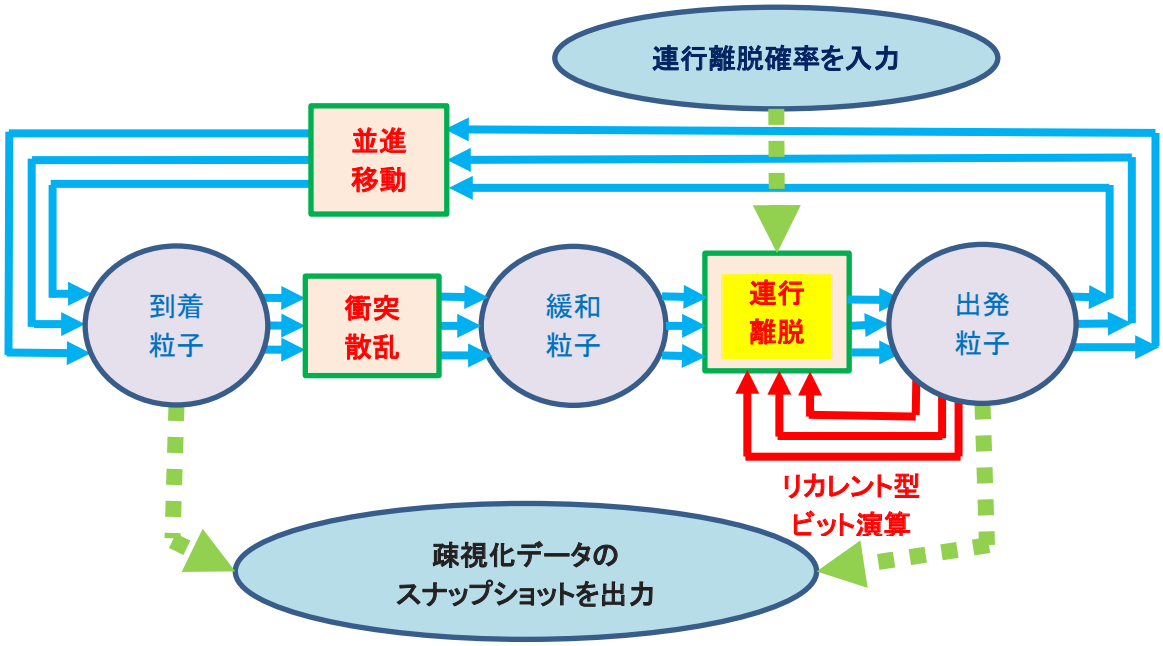


図２．“連行離脱過程”を組み込んだ格子ガス法の計算手順

（２）“連行離脱過程”を実現するビット演算プログラム

ここでは、「ある向きへの連行過程」が生じる場合は、①“静止仮想粒子”が“同じ向きに動く運動仮想粒子”になるか、②“逆向きに動く運動仮想粒子”が“静止仮想粒子”になると考える。また、「ある向きからの離脱過程」が生じる場合は、①“同じ向きに動く運動粒子”が“静止仮想粒子”になるか、②“静止仮想粒子”が“逆向きに動く運動仮想粒子”になると考える。このとき必要になる運動量とエネルギーのやりとりは、前述のとおり、「“静止仮想粒子”が“仮想光子場”から運動量とエネルギーをもらい“運動仮想粒子”になる過程」と「“運動仮想粒子”から“仮想光子”が分離して“仮想光子場”に運動量とエネルギーを返却して“静止仮想粒子”になる過程」を通じて行う。より具体的には、いくつかの方法が考えられるが、筆者が現時点で一番簡単で計算負荷の少ない方法だと考えている方法を次に示す。従来の通常の格子ガス法モデルのイメージとはかなり異なっており、その主要な特徴は次の４つである。

【特徴１】格子点間を移動中の仮想粒子集団について、重心運動に伴う運動量とエネルギーの有無のみに着目する。

⇒各格子点とその隣接格子点の間は、いつでもどこでもほぼ同じ数密度の多数の仮想粒子によって埋め尽くされていると仮定する。これは、通常の数値流体力学における“非圧縮性流体”の仮定に対応している。この格子点間を移動中の仮想粒子集団について、「個々の仮想粒子は複雑にぶつかり合っているかもしれないが、全体集団としては、ある格子点から移動先の格子点に向かって円滑に流れている状態」を“１”（快走）、「個々の仮想粒子は複雑にぶつかり合っているが、全体集団としての移動は停滞している状態」を“０”（渋滞）と考える。この場合の「１」と「０」の概念は、仮想粒子集団がもつ“重心の運動量”と“重心の運動エネルギー”の有無のみに着目したものと言える。“質量”については、「１」の場合も「０」の場合も、格子点間に同じ量が存在しているイメージであり、区別する必要がないので考察対象からははずすことにする。

【特徴２】各格子点における“衝突散乱過程”が時刻０から時間間隔⊿τごとに生じていると仮定した場合、「時刻 t に生じた“衝突散乱過程”の直後に“連行離脱過程”を続けて生じさせるか否か？」の判断を、時刻 $t-\frac{\Delta\tau}{2}$ における最近傍にいる仮想粒子集団の運動状態（“快走”または“渋滞”）によって判断する。

⇒格子ガス法で周辺の状況を考慮する通常の方法では、時刻 t において注目している格子点のすべての最隣接格子点について、それぞれの格子点に存在するすべての速度(速さと向き)の仮想粒子の有無に関する情報を集計する必要がある。ここで採用しているテシャラの４次元 FCHC モデル[4]の場合、運動仮想粒子の速度は 48 種類であるので、48×48＝2304 個の仮想粒子の状態情報(存在 or 不在)を集めることが必要になる。これに対し、ここで提案する方法では、時刻 $t-\frac{\Delta\tau}{2}$ において最近傍にいる仮想粒子集団は最大でも 48 個であり、48 個の仮想粒子集団の状態情報(快走 or 渋滞)を調べればよいことになって、計算負荷を軽減できる。なお、静止仮想粒子集団は、格子点間を移動できず、運動量がゼロなので、その状態情報を集める必要はない。

【特徴３】「“連行過程”または“離脱過程”を生じさせるか否か？」の具体的な判断は、ひとつの方向(互いに逆の向きをなすふたつの速度の対：**D**と$\overline{\mathbf{D}}$)ごとに独立に判断する。

【特徴４】今回は、一番簡単で計算負荷の少ない方法を試すという趣旨から、“離脱過程”を考慮せず、“連行過程”のみを制御してみることにする。

今回用いた“連行過程”の制御を実現するプログラムを**表１**に、アルゴリズムを**図３**に示す。

表１．“連行過程”を実現するビット演算プログラム（１方向）

プログラム：

```
AF = AR[D];  DF = DP[D];  RF = RX[D];  PF = PH[D];
AB = AR[D̄];  DB = DP[D̄]; RB = RX[D̄];  PB = PH[D̄];
FCH = (~AF & ~AB &  DF & ~DB & ~RF &  ZR &  PF & RENKO)
     |(~AF & ~AB & ~DF &  DB &  RF & ~ZR & ~PF & RENKO);
BCH = (~AF & ~AB & ~DF &  DB & ~RB &  ZR &  PB & RENKO)
     |(~AF & ~AB &  DF & ~DB &  RB & ~ZB & ~PB & RENKO);
ZR  = ZR ^ (FUP | BUP);
RX[D] = RF ^ FUP;   RX[D̄] = RB ^ BUP;
PH[D] = PF ^ FUP;   PH[D̄] = PB ^ BUP;
```

ここで、

AR[D]： 時刻 t にＤ向きの速度をもって到着した仮想粒子集団の存否配列
DP[D]： 時刻 t -1 にＤ向きの速度をもって出発した仮想粒子集団の存否配列
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集団の存否配列
PH[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光子集団の存否配列
AR[$\overline{\mathrm{D}}$]： 時刻 t に逆Ｄ向きの速度をもって到着した仮想粒子集団の存否配列
DP[$\overline{\mathrm{D}}$]： 時刻 t -1 に逆Ｄ向きの速度をもって出発した仮想粒子集団の存否配列
RX[$\overline{\mathrm{D}}$]： 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集団の存否配列
PH[$\overline{\mathrm{D}}$]： 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光子集団の存否配列
ZR　： 時刻 t に格子点上で静止している仮想粒子集団の存否配列
RENKO： 連行過程を試みる確率に応じて、RENKO の各ビットの「１」の出現数を決める。
「&」、「|」、「^」、「~」：ビット演算の「論理積」、「論理和」、「排他的論理和」、「論理否定」

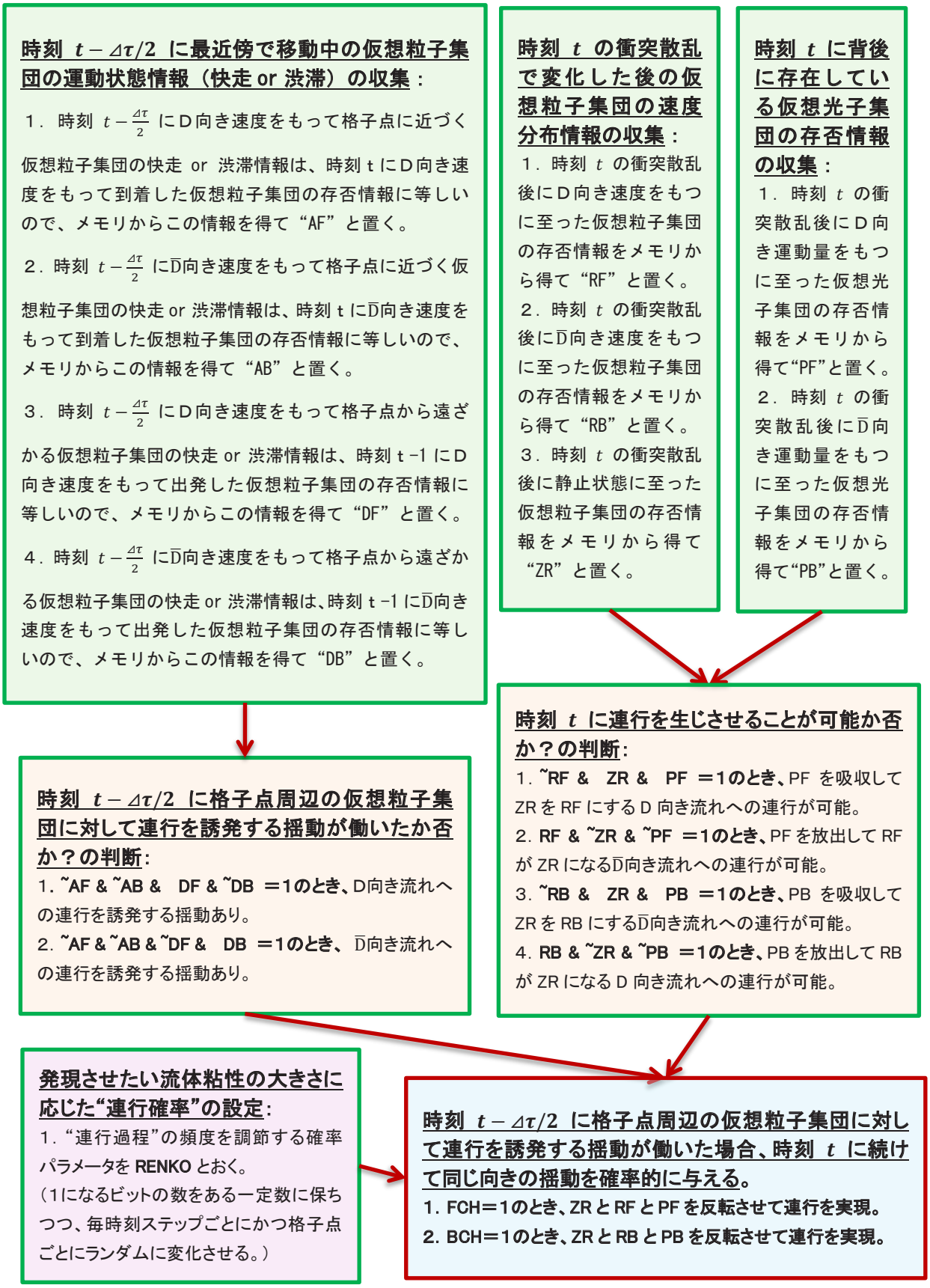


図３．粘性制御のための“連行過程”を実現する計算アルゴリズム

３．“仮想光子場揺動モデル”に基づく四角柱の後流解析

（１）　計算体系

はじめに、各辺がＸ、Ｙ、Ｚ方向を向く直方体体系の静止流体を考える。その中に中心軸がＺ方向をもつ四角柱（アスペクト比＝0.125 の四角柱）を置く。時刻ステップ 0 において、流体が＋Ｘ向きに突然流れ出した場合に、四角柱の後流がどのように変化するか？を数値シミュレーションにより求める。今回の計算の特徴は、Ｘ方向の計算体系を非常に長く設定したことである。四角柱の横幅の 100 倍以上遠方までのマルチスケールな範囲で変化する後流の挙動を計算した。

計算体系における格子点配置は、Ｘ方向 8192×Ｙ方向 2048×Ｚ方向 16 ≒2.68 億個の格子点である。“４次元面心超立方体(FCHC)格子”を採用しているため、個々の３次元格子点には、４次元目の位置の自由度がある。今回のモデルでは、超単純化をめざすため、４次元目の位置の自由度は最小値である「２」に設定した。

Ｚ軸に平行に配置する四角柱の中心軸は、流体の注入口(Ｘ＝0)から下流に 512 格子点の位置に置いた。四角柱の“Ｙ方向の横幅”は、64 個の格子点が並ぶ長さとし、この長さに“アスペクト比”を乗じた長さを四角柱の“Ｘ方向の厚さ”とした。Ｙ方向とＺ方向には、周期境界条件を適用するので、四角柱のＺ方向の長さは無限大ということになる。以上の状況を**図４**に示す。

今回、最終的に求める物理量は、各疎視化位置における“流体の運動量ベクトル”である。４×４×４＝64 個の３次元格子点（すなわち、４×４×４×２＝128 個の４次元格子点）からなる“疎視化セル”ごとに、個々の仮想粒子がもつ運動量ベクトルを合計する方法で算出した。

時間発展計算は、時刻ステップ 0 から始めて 38400 時刻ステップまでの計算を行い、256 時刻ステップを経過するたびにスナップショットを出力させた。従って、スナップショットの総数は、初期画面(No.0)から No.150 の画面までの 151 枚である。スナップショットで出力されるデータは、四角柱の中央で、その軸に垂直に四角柱を切る平面を考え、その平面上に位置する“疎視化セル”がもつ運動量ベクトルである。なお、可視化には、ParaView を用いた。

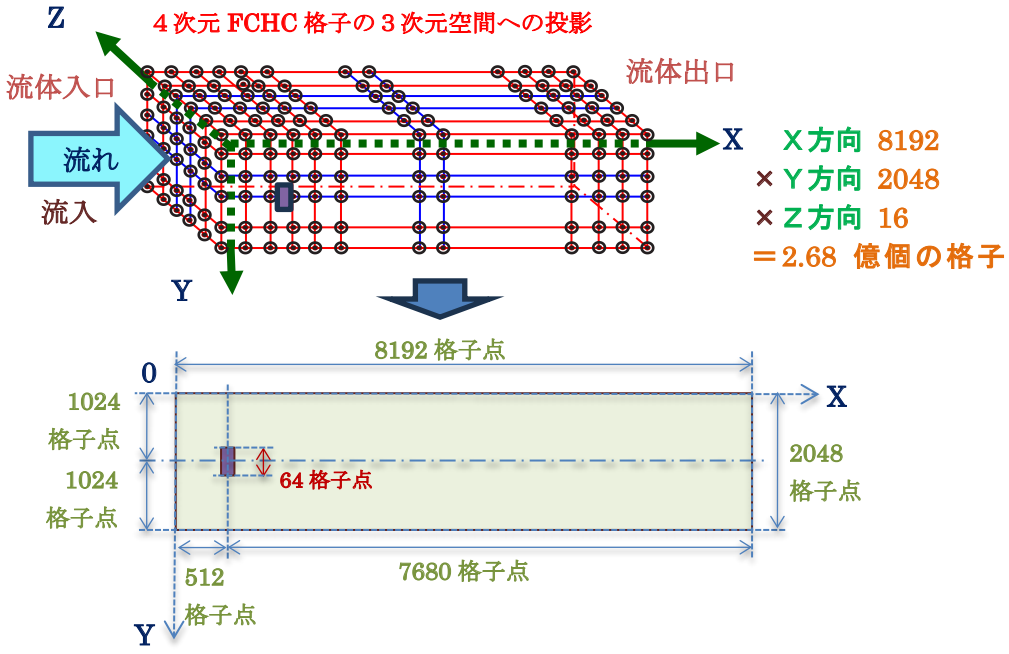


図４．四角柱の遠方後流までの解析を行うための格子点体系

（2）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平均でゼロであり、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 に位置する流体入口から注入していく。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側の先にある流体出口においては、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。

（3）計算結果

図 5 に、アスペクト比を 0.125 にした場合の四角柱の後流における流速と Y 成分速度の過渡変化を示す。スナップショット No. 30, 50, 70, 90, 110, 130, 150 を上から順に並べて過渡変化を示した。左側の列が流速の大きさの過渡変化であり、右側の列が流速のY方向成分の過渡変化である。なお、実際の可視化では、ParaView の機能により、151 枚のスナップショットを、動画の形で直観的に観察できる。なお、粘性を制御する“連行確率”は、0.2%とした。

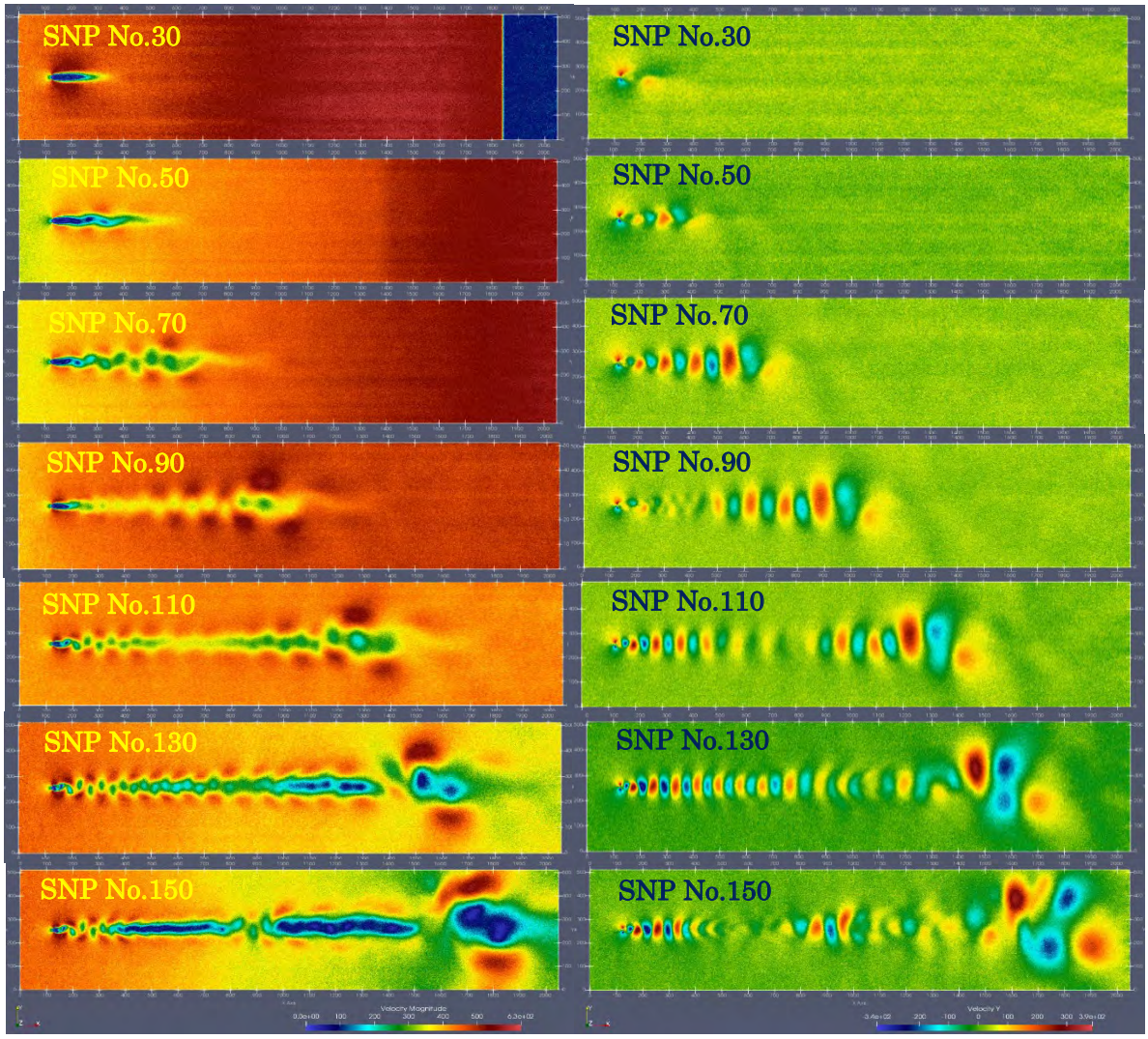


図 5．四角柱（アスペクト比 0.125）の後流における流速と Y 成分速度の過渡変化

図５の一番下に示した最後のスナップショット No. 150 を拡大して**図６**に示す。そこには、X 成分速度の分布も追記してある。なお、図中の A～E の区画は、次節の説明のために付記した。

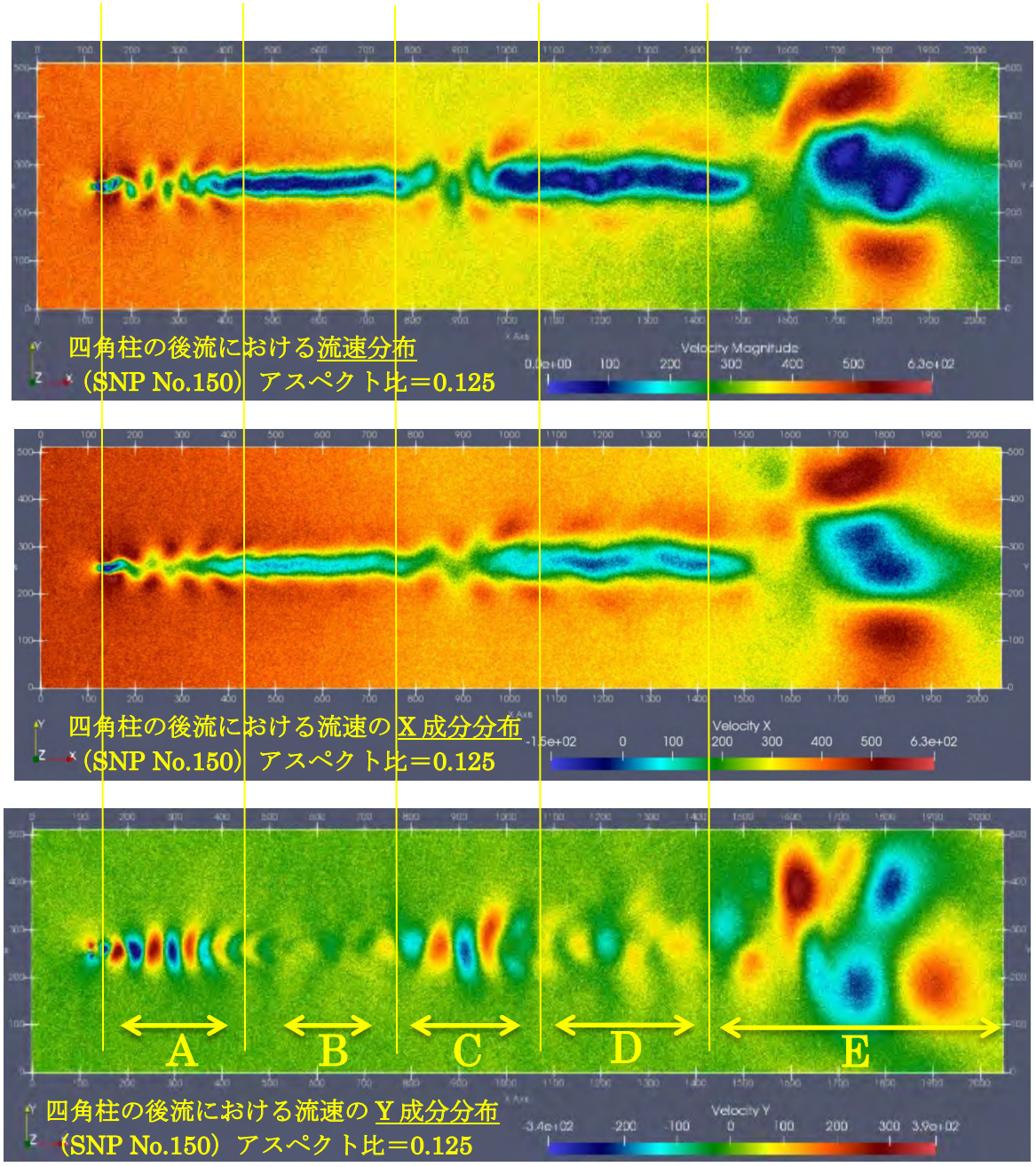


図６．四角柱（アスペクト比 0.125）の後流における
時刻ステップ No. 150 時点の流速及び Y 成分速度並びに特徴の領域分け

以上の計算には、東北大学サイバーサイエンスセンターの AOBA－S を使用し、１ケース（上述のとおり、2.68 億個の 3 次元格子点についての時間発展計算を 38400 時刻ステップ行う計算）を、「16 コア/VE×8VE」のベクトルプロセッサによる 128MPI 並列計算で約３時間 20 分であった。（ベクトル化率：約 98.8%、ベクトル長：約 220）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。

（4） 結果の考察

図６の一番下の図に示したＹ成分速度の分布をみると、四角柱の後流は、下流向ってＡ～Ｅの５領域に分けることができる。このうち、A, C, E の領域では、Ｙ成分速度が正負に激しく振動する“カルマン渦列”が形成されている。Ｂと Ｄの領域では、Ｙ成分速度の振動が弱まり、カルマン渦列がほとんど崩壊している。また、カルマン渦列が存在する領域のＹ方向の後流幅は、Ａ ⇒ Ｃ ⇒ Ｅの順に広くなっている。

次に、これらの知見を、先行研究の知見と定性的に比較してみる。
なお、柱状物体の遠方後流までのカルマン渦列の挙動については、多数の先行研究がある。ここでは、紙面の都合もあり、特にわかりやすいものだけとりあげる。

“カルマン渦列の崩壊と再配列”については、文献[5]に流体の可視化実験等の結果とそれに基づく考察が多数記載されている。特に、その 49 節（P103）には、以下の知見が述べられている。

【以下、引用部分】----------
「中高レイノルズ数においては、柱状物体の後方に例外なくカルマン渦列が形成されるが、カルマン渦列は一定の配列を長い距離にわたって保つことができない。物体から離れるにつれて形状を変え続け、ついには崩壊する。一般にレイノルズ数が大きいほど崩壊する位置は物体に近づく。カルマン渦列が崩壊したあと、後流はしばらくの間乱雑な状態を続けるが、その後方で再びきれいなカルマン渦列を形成することが多い。２回目に出現するカルマン渦列の大きさは最初のカルマン渦列より数倍大きい。２回目に現われたカルマン渦列も下流では再び崩壊する。このようにして、後流は下流方向にカルマン渦列の形成と崩壊を繰り返す。
（・・・中略・・・）
この興味深い現象は、カルマン渦列が後流の幅に比例する波長をもつ不安定波であると考えれば説明できる。後流の幅は粘性の作用により下流へ向かって増大するから、後流中には下流ほど大きな波長をもつカルマン渦列が次々と成長するものと思われる。事実、実測されたカルマン渦列の波長は常にその場所における後流幅にだいたい等しいのである。」
----------（引用部分終了）

上記の引用文献における実験結果は、主に、静水槽中を一定速度で進行する円柱等の実験から得られた知見である。これに対し、今回行った数値シミュレーションは、静止した四角柱の周囲にある流体を突然動かしはじめたときの過渡変化であり、直接比較することはできない。しかし、両者は、相対運動として似たような現象であることから、定性的には共通の知見が得られるものと期待される。この観点から、上記記載の下線部で示した２つの知見：
・後流は下流方向にカルマン渦列の形成と崩壊を繰り返す。
・実測されたカルマン渦列の波長は常にその場所における後流幅にだいたい等しい。
について、**図６**の結果は、整合性のある結果であることを示している。

また、**図６**の Ｂと Ｄの領域は、“平行二重渦層”に近い流れになっている。四角柱ではなく円柱後流に関する実験結果からではあるが、「カルマン渦列の崩壊と二次渦列の生成過程は、渦度の流れ方向への拡散により、カルマン渦列が平行二重渦層的流れに移行し、その流れの不安定性により二次渦列が生成されると解釈できる。」とした先行研究[6]があり、この知見と**図６**の結果には整合性があることがわかる。

さらに、四角柱のアスペクト比を 0.5 にして全く同じ計算を行った場合の結果を**図７**に示す。

これを、アスペクト比が 0.125 の場合の A～E の 5 領域の位置（**図 6** 一番下の図）と比較すると、少なくとも、アスペクト比が 0.125 の場合の第 1 渦列が崩壊する場所（領域 A の右端）は、アスペクト比が 0.5 の場合に崩壊する場所よりも上流側に位置していることがわかる。この傾向は、先行研究の数値シミュレーションによる結果[7]などと整合している。

ただし、数値シミュレーションの条件が先行研究の場合とは多少異なっているので、正確な比較はできない。特に、時刻ステップ No. 150 の時点における後流の状態は、先行研究が解析対象にしている定常状態にはまだ完全には達していないと考えられる。また、先行研究では、通常の CFD による 2 次元非圧縮性流体解析によるものが多く、ここでの解析は、格子ガス法に粘性制御機能を付加した 3 次元計算（ただし、Z 方向が極端に短い直方体体系）であった。このため、今回の比較では、定性的な傾向の整合性についての考察に留めた。

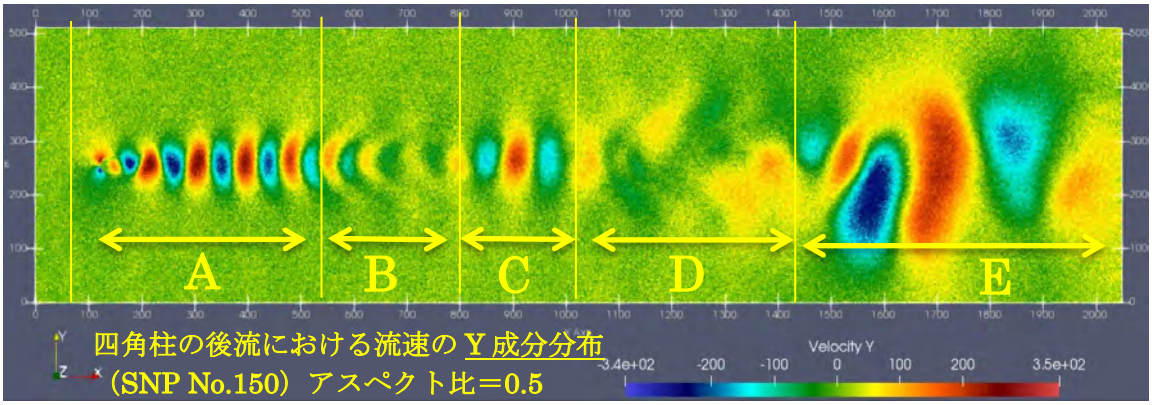


図 7．四角柱（アスペクト比 0.5）の後流における
カルマン渦列の Y 成分と特徴の領域分け

4．本計算アルゴリズムの応用可能性

今回、**図 4** に示したとおり、四角柱の横幅長さに 64 個の格子点を配置して、そこからこの横幅長さの 100 倍以上離れた下流までを解析対象として数値シミュレーション行った。その結果、柱状物体の後流挙動に特徴的なカルマン渦列の形成と崩壊の繰り返しを**図 6** や**図 7** のとおり再現できた。これが、通常の CFD シミュレーションと比較して正確なものであるか否かの検証はできていないが、少なくとも定性的には整合性のある結果を得た。

図 3 に示した粘性制御のための“連行過程”を実現する計算アルゴリズムについては、“連行確率”をゼロにすると、カルマン渦列の形成自体が生じないことを確認している。すなわち、粘性が大きい状態である。他方、今回の数値シミュレーションでは“連行確率”を 0.2%としたが、この値を大きくしていくと、あるところで負の粘性状態が発現し、静止流体からも自発的な渦発生が生じることを確認している。従って、連行確率をうまく制御してやれば、粘性係数が正数値で 0 に非常に近い状態を発現でき、非常に大きなレイノルズ数のシミュレーションが可能になると思われる。このとき、流体がどんなに激しく変動する乱流状態になったとしても、ビット演算（＋シフト演算）のみで時間発展計算を実行できる格子ガス法では、数値計算上の不安定性はない。

さらに、今回示した「粘性制御のための“連行過程”を実現する計算アルゴリズム」は、第 1 章で述べたように、基本的には、非平衡統計力学における流体粘性モデルの発想をそのまま格子ガス法の計算モデルに組み込んだ形のものであり、それなりの合理性がある。

以上のことから、“仮想光子場揺動モデル”についての今後の応用可能性が期待できる。

謝辞

本研究では、本文にも述べたとおり、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-S(SX-Aurora TSUBASA)の「16 コア/VE×8VE」を利用して 2.68 億格子点の時間発展計算 38400 時刻ステップを約３時間 20 分で実行した。これは、非常にコンパクトに作られた C401-8 モデル１台の規模であり、AOBA-S には、504 台の C401-8 モデルが存在するので、1000 億格子点を超える規模の計算続行も、計算費用さえ気にしなければ十分実行可能であることがわかる。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開発導入と円滑な運用、さらにご親切なユーザー支援活動に心から感謝する次第である。

参考文献

[1] 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

[2] 松岡, “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”, SENAC Vol.57 No.2, pp.15-24, 2024

[3] 香取,“非平衡統計力学”, pp89-127, 1999, 裳華房 (ISBN 978-4-7853-2086-7)

[4] Christopher M. Teixeira, “Continuum Limit of Lattice Gas Fluid Dynamics”, Ph.D. Thesis, MIT, 1993

[5] 種子田,“画像から学ぶ流体力学”, 朝倉書店, 1988, ISBN 4-254-13040-6 C 3042 (cf. Taneda, “Downstream development of the wakes behind cylinders”, J. Phys. Soc. Japan 14 (1959) 84)

[6] 烏谷, 船越, 星野, “円柱後流渦度分布の変化”, 九州大学応用力学研究所報　第 66 号, pp.187-193, 1988

[7] Inasawa A., Asai M. and Nakano T.: Sound generation in the flow behind a rectangular cylinder of various aspect ratios at low Mach numbers, Computers and Fluids, 82, 148-157, 2013

［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「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-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 5.6PB、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。

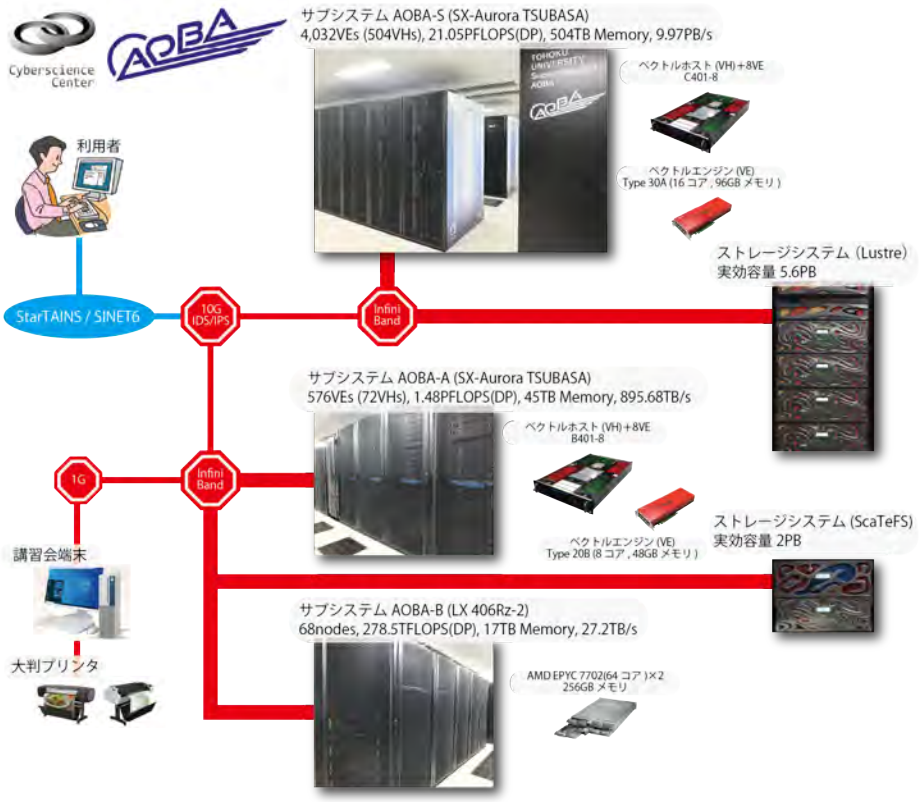


図 1　スーパーコンピュータ AOBA の構成

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。

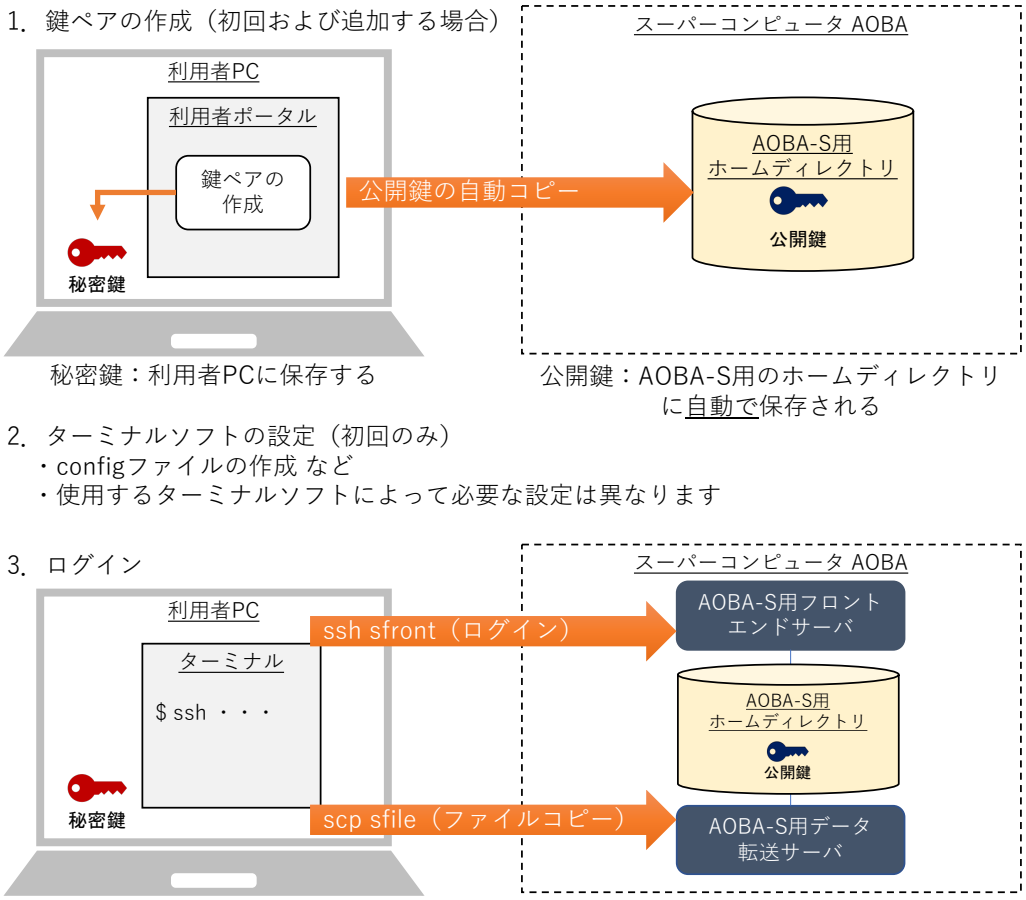


図 3　鍵ペアの作成からログインまで

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。

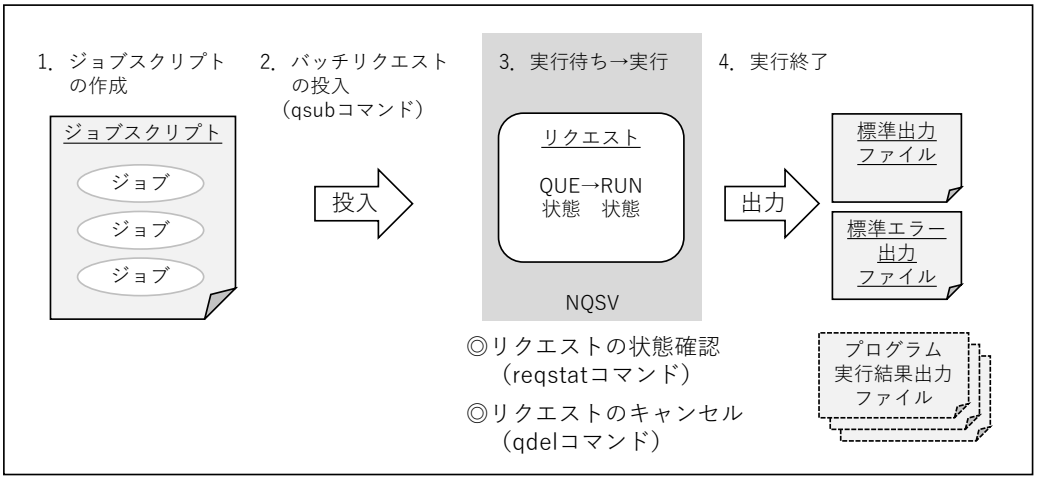


図 5　バッチリクエストの概念図

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/

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。

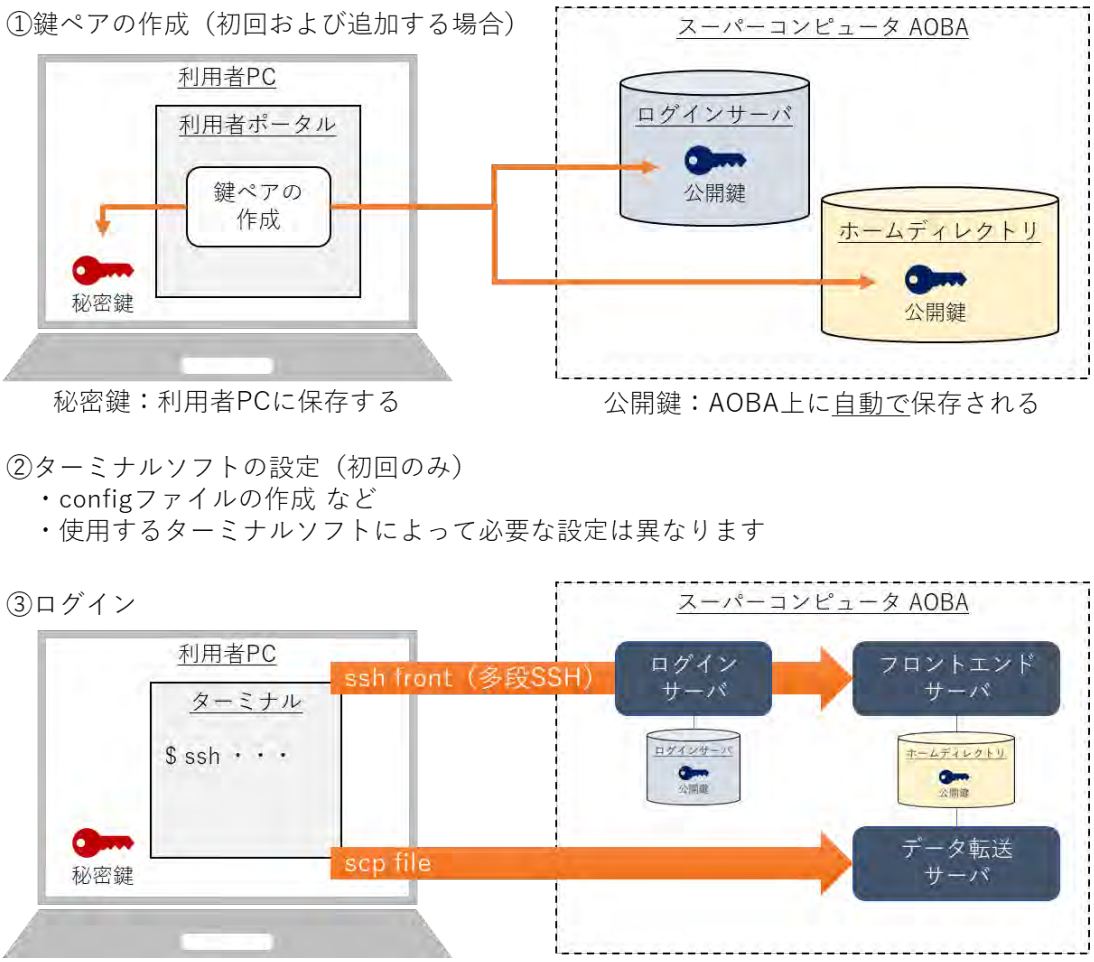


図 1　鍵ペア作成からログインまでの流れ

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-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 5.6PB、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。

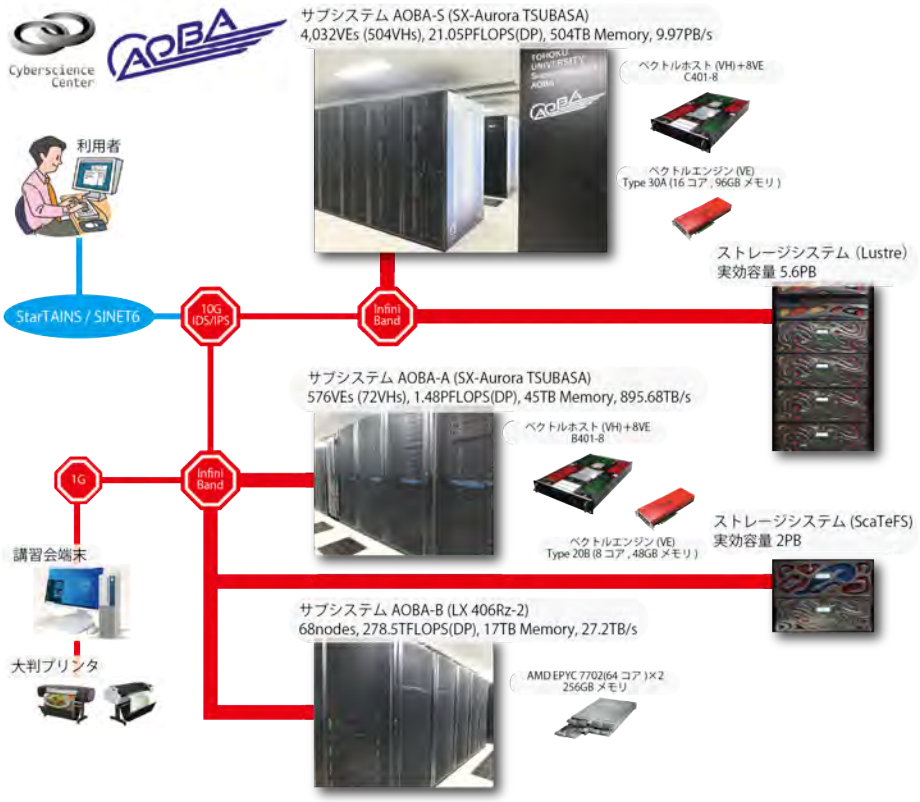


図 1　スーパーコンピュータ AOBA の構成

2.1 サブシステム AOBA-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。

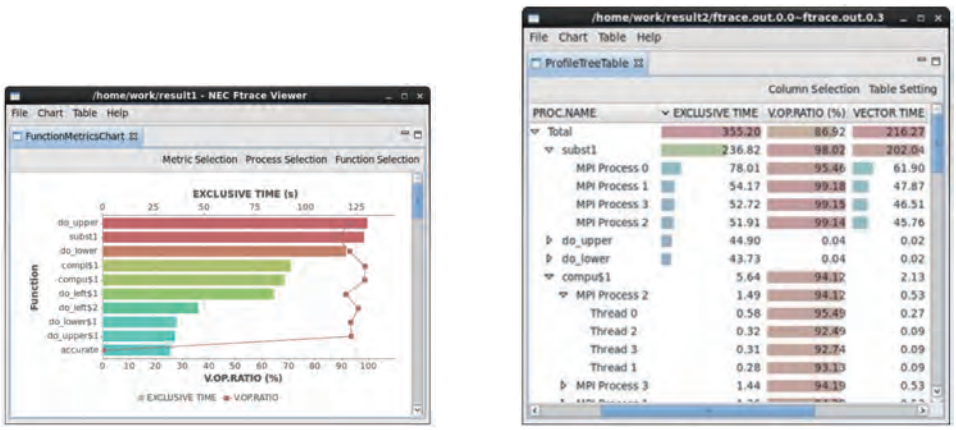


図 4　Ftrace Viewer の表示例

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。

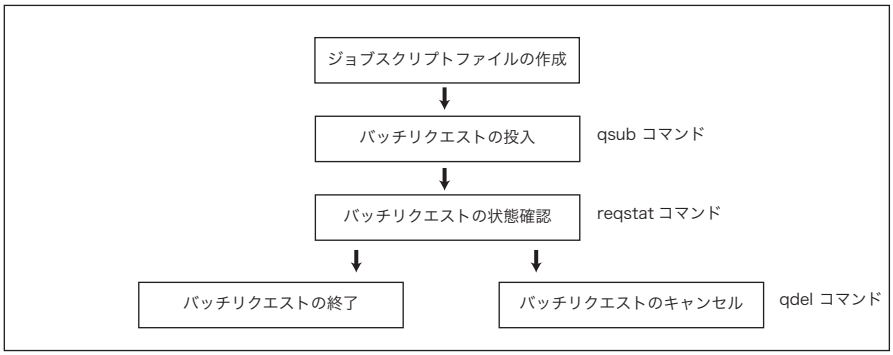


図 5　バッチリクエストの概念図

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.sh の記述内容）
#!/bin/sh
#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.sh の記述内容）
#!/bin/sh
#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.sh の記述内容）
#!/bin/sh
#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.sh の記述内容）
#!/bin/sh
#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.sh の記述内容）
#!/bin/sh
#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.sh の記述内容）
#!/bin/sh
#PBS -q sx #AOBA-Aを使用する
#PBS --venode 8 #VEを8個使う
#PBS -l elapstim_req=2:00:00 #最大経過時間を2時間に指定
#PBS -v NMPI_SEPSELECT=3 #出力形式に3を指定

cd $PBS_O_WORKDIR #qsubを実行したディレクトリに移動
mpirun -np 64 /opt/nec/ve/bin/mpisep.sh ./a.out #カレントディレクトリの a.out を64プロセス並列で実行
```

表 5　NMPI_SEPSELECT の引数と動作

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

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

4　スーパーコンピュータ用数値演算ライブラリ

4.1　NLC (NEC Numeric Library Collection)

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

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

4.1.1　Fortran プログラムから利用する場合

Fortran プログラムから利用する場合，表 6 に示すライブラリが使用できます。

表 6　Fortran 用 NLC ライブラリと機能概要

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

4.1.2　C プログラムから利用する場合

C プログラムから利用する場合，表 7 に示すライブラリが使用できます。

表 7　C 言語用 NLC ライブラリと機能概要

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

NLC ライブラリの詳細およびコンパイルとリンク方法については 5 章の NLC (NEC Numeric Library Collection) ユーザーズガイドをご参照ください。

5　マニュアル

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

https://www.hpc.nec/documentation

当センター独自の運用方法により，マニュアルに記載の事項が動作しない場合もあります。

5.1 コンパイラマニュアル

コンパイラについては以下のマニュアルをご参照下さい。

- C/C++ Compiler ユーザーズガイド, C/C++ Compiler User's Guide
- Fortran Compiler ユーザーズガイド, Fortran Compiler User's Guide
- NEC MPI ユーザーズガイド, NEC MPI User's Guide

5.2 性能情報取得についてのマニュアル

実行時の性能情報取得については以下のマニュアルをご参照下さい。

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

5.3 数値計算ライブラリ（NLC）マニュアル

数値計算ライブラリ（NLC）については以下のマニュアルをご参照下さい。

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

6 おわりに

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

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

［大規模科学計算システム］【AOBA-A および AOBA-B の利用法】

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-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 5.6PB、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。

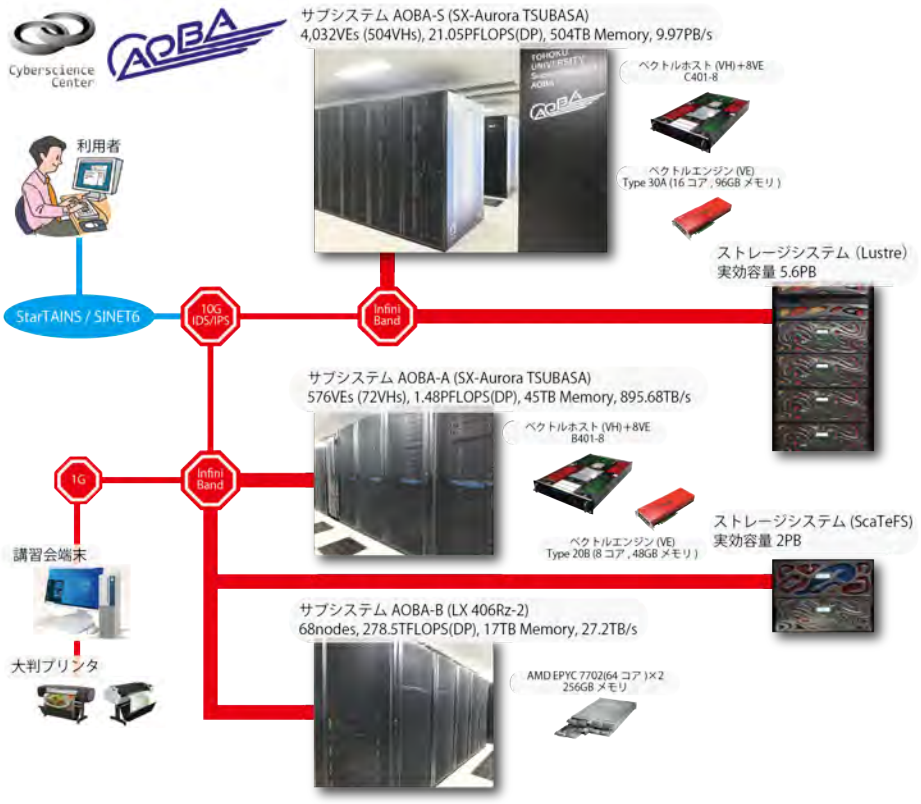


図 1　スーパーコンピュータ AOBA の構成

2.1 サブシステム AOBA-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。

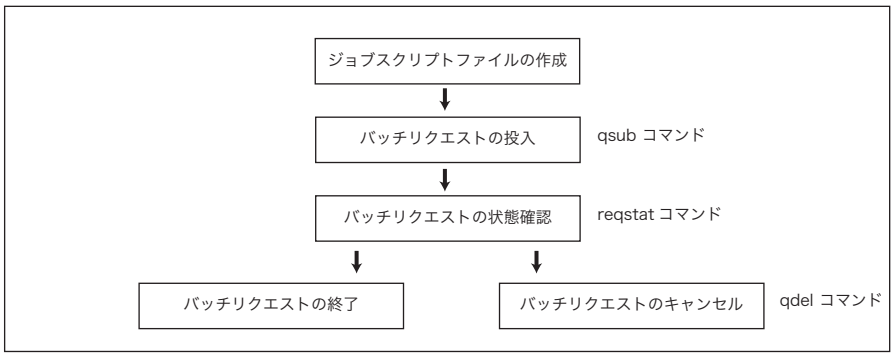


図 4　バッチリクエストの概念図

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.sh の記述内容）
#!/bin/sh
#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.sh の記述内容）
#!/bin/sh
#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.sh の記述内容）
#!/bin/sh
#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.sh の記述内容）
#!/bin/sh
#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.

1 つ目に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について説明します.

2 つ目に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．
  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．

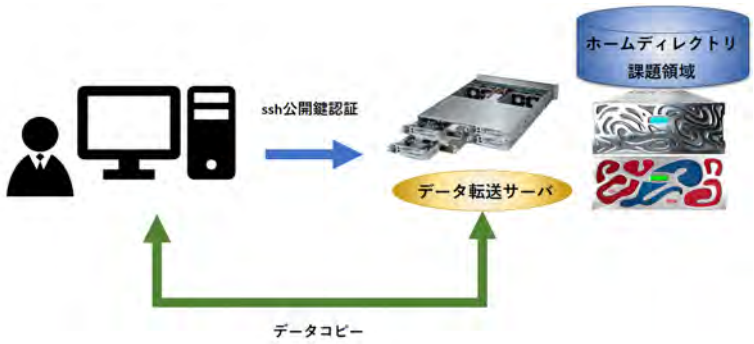


図 1　アクセスイメージ

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．

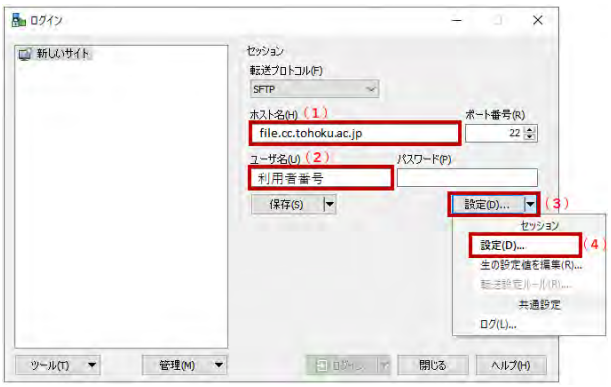


図 2　WinSCP 設定画面

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.

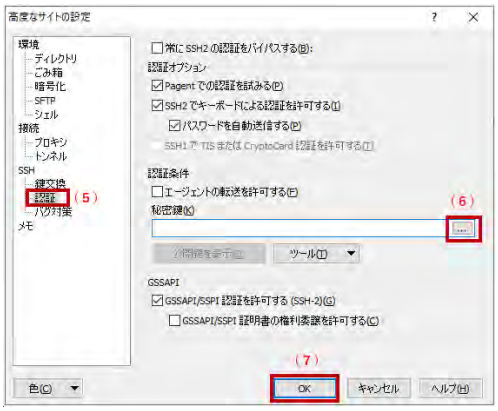


図 3　WinSCP 鍵認証設定画面

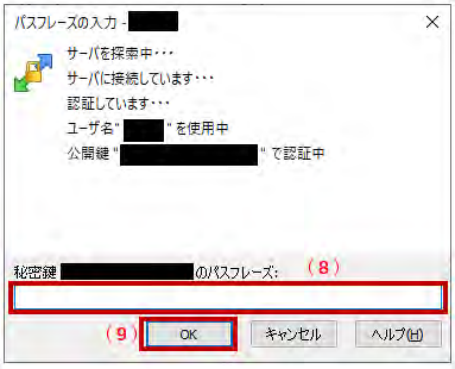


図 4　WinSCP 鍵認証画面

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.

［大規模科学計算システム］【AOBA-A および AOBA-B の利用法】

大判カラープリンタの利用法

情報部デジタルサービス支援課

1　はじめに

センターでは 2020 年 10 月から新大判カラープリンタシステム（ヒューレットパッカード社、HP DesignJet Z6 PS dr V-Trimmer）を導入しました。

光沢紙（600 円／枚）またはソフトクロス紙（1,200 円／枚）に、A0 サイズまでの印刷が可能です。従来の大判カラープリンタでは印刷後に左右の余白をカットする必要がありましたが、今回導入した大判カラープリンタには自動的に左右の余白をカットする機能 V-Trimmer が搭載されており、余白のカットの手間がなくなりました。

大判カラープリンタは、サイバーサイエンスセンター本館 1 階の利用相談室にあります。利用時間は平日 9 時～21 時です。大判カラープリンタを利用するためには、利用負担金が発生しますのでサイバーサイエンスセンターの利用者番号が必要です。

本稿ではセンター利用相談室の Windows 端末から PowerPoint や PDF などのアプリケーションデータを大判カラープリンタで印刷する方法について説明します。

図 1　大判カラープリンタシステムの構成 (2020 年 10 月 1 日～)

2　利用負担金

光沢紙への印刷は、1 枚（1 部）あたり 600 円、ソフトクロス紙への印刷は、1 枚（1 部）あたり 1,200 円の利用負担金が発生します。いずれの利用負担金も 1 枚（1 部）あたりの金額ですので、出力サイズの大小によりません。

3　印刷に必要なもの

■ サイバーサイエンスセンターの利用者番号　センターに利用登録をしていない場合は、利用申請が必要です。利用者番号の取得までは平日 3 日間程度かかります。日数に余裕を持って利用申請を行ってください。
「利用申請からログインまで」　https://www.ss.cc.tohoku.ac.jp/first-use/

■ 印刷データ　利用者の PC / Mac 等で作成したポスター用に作成したデータ（パワーポイント、PDF）を USB メモリ等に入れてお持ちください。PDF を作成する場合は、フォントを全て埋め込んでください。Mac で作成したパワーポイントのファイルや、Google スライドで作成した PDF ファイルは、フォント不足やレイアウトが崩れることがあります。

4　印刷手順

4.1　Windows 端末へのログイン

利用相談室の 3 台の Windows 端末から印刷が可能です（図 2）。
サイバーサイエンスセンターの利用者番号と、LDAP パスワード（初期パスワードは利用承認書に記載のもの）で Windows にログインしてください（図 3）。

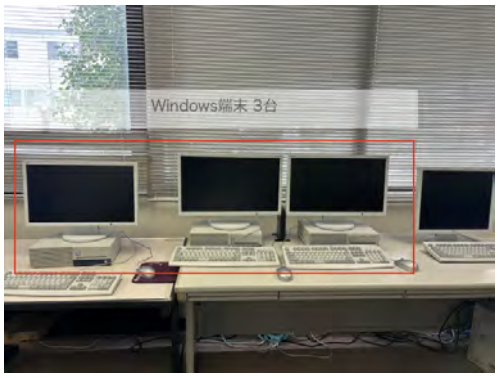


図 2　Windows 端末

図 3　ログイン画面

4.2　印刷データを開く

Windows 端末の USB ポートに USB メモリ等を接続し、印刷データを開きます。各アプリケーションの印刷メニューを選択してください。PDF ファイルから印刷する場合は、Adobe Acrobat Reader もしくは Microsoft Edge Browser が利用できます。Google スライドで作成した PDF ファイルはレイアウト崩れが発生するため、Microsoft Edge Browser から印刷を行ってください。

4.3　プリンタの選択

1. 使用するプリンタを選択します（図 4)。「Z6dr 44in Left」と「Z6dr 44in Right」のどちらのプリンタを選択しても、光沢紙とソフトクロス紙での印刷が可能です。
2. 印刷する用紙サイズ等を設定するために、「プロパティ（P)」を選択します。

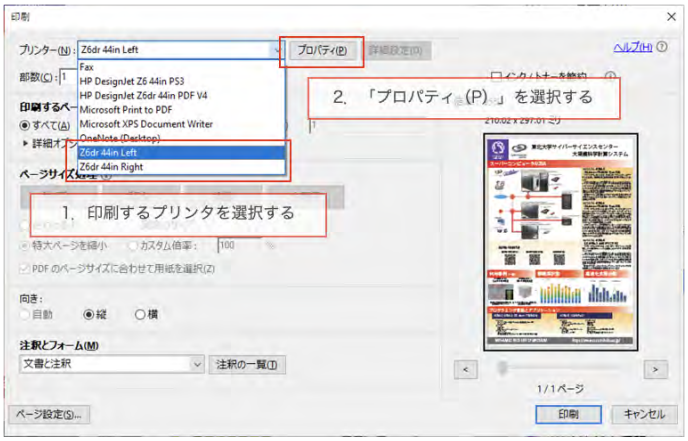


図 4　印刷メニュー

4.4　印刷用紙の選択

それぞれのプリンタには 2 種類の印刷用紙が取り付けられています。印刷する用紙に合わせて給紙方法を選択してください（図 5)。

■ **光沢紙に印刷する場合**　「ロール紙 1」を選択

■ **ソフトクロス紙に印刷する場合**　「ロール紙 2」を選択

図 5　印刷用紙の選択

4.5　印刷品質の変更

必要な場合は印刷品質を変更してください（図 6）。

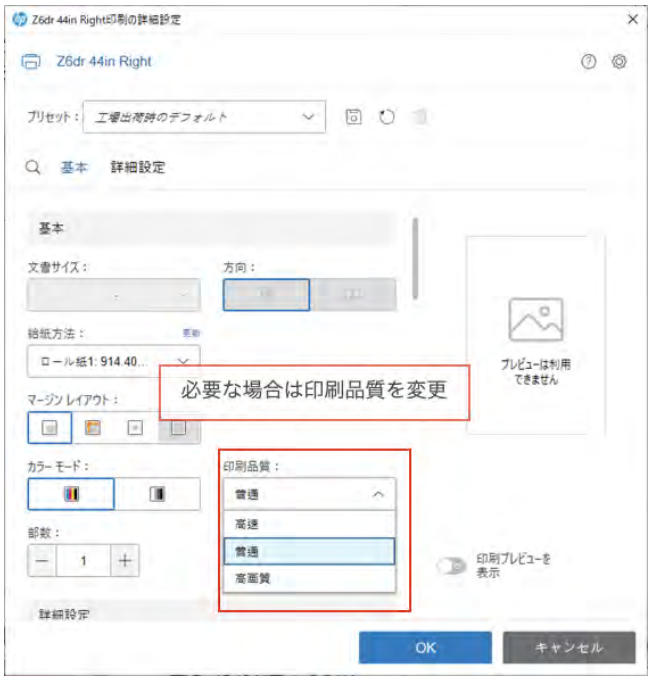


図 6　印刷品質の変更

4.6　自動カッター機能を有効にする

垂直トリマー（縦方向の自動カッター機能）をオンにしてください（図 7）。

図 7　自動カッター機能

4.7　出力サイズの選択

「スケール」で出力サイズを選択してください。A0 サイズまで出力可能です。プリンタ名と出力サイズ、その他設定を確認後、「OK」を押してください（図 8）。

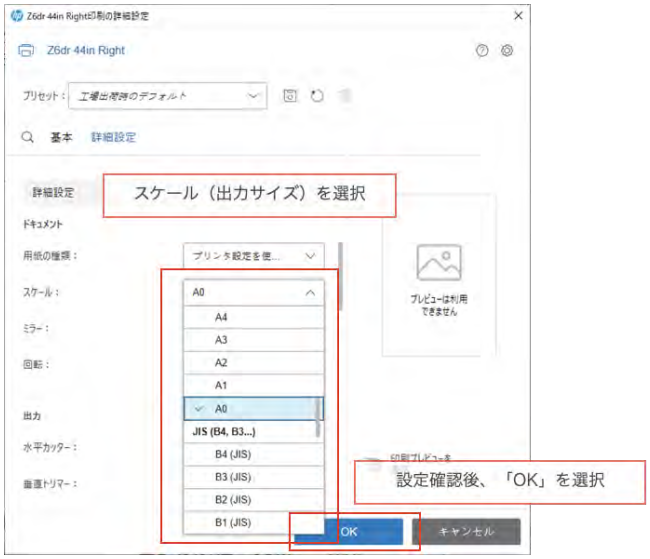


図 8　出力サイズの選択

4.8　印刷開始

「印刷」を選択すると印刷を開始します（図 9）。

光沢紙への印刷の場合乾燥時間が必要なため、印刷完了後にプリンタが数分停止します。印刷が完了すると自動的に切断されますので、**絶対に用紙を引っ張らない**でください。プリンタ故障の原因になります。

図 9　印刷メニュー

5　その他

- 印刷で出た用紙の切れ端は、利用相談室内のゴミ箱に捨ててください。
- 印刷用紙が無くなった場合は、利用者での用紙交換が可能です。利用相談室内のロール紙をご利用ください。
- 印刷終了後は Windows 端末をシャットダウンしてください。
- 使用した USB メモリ等は忘れずにお持ち帰りください。
- 最後の退出者の方は、電灯とエアコンの電源を OFF にしてください。
- 利用について問い合わせがある場合は、利用相談室の電話機から内線 832 または 829 におかけください。担当者不在の場合は当日の対応が出来ませんのでご了承ください。

6　おわりに

今回センターに導入した大判カラープリンタは V-Trimmer 機能により、余白のカットの手間がなくなりました。プリンターで印刷可能なソフトクロス紙は、織り目がつきにくく持ち運びに非常に便利です。また、光沢紙は色鮮やかな印刷が可能です。研究成果のポスター印刷にぜひご活用ください。

［お知らせ］

2025 年度サイバーサイエンスセンター講習会のご案内

No.	講習会名 開催方式	開催日時	募集人員	講師	内容
1	はじめての Linux オンライン	5 月 21 日(水) 13:00-15:00	15	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 オンライン	5 月 23 日(金) 13:00-15:00	15	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	ネットワークとセキュリティ入門 オンライン	8 月 4 日(月) 13:30-15:30	90	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
4	はじめての Linux オンライン	9 月 9 日(火) 13:00-15:00	15	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
5	並列プログラミング入門Ⅰ（OpenMP） オンライン	9 月 11 日(木) 13:00-16:00	15	河合 (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
6	並列プログラミング入門Ⅱ（MPI） オンライン	9 月 16 日(火) 13:00-16:00	15	担当者 (NEC)	・MPI による並列プログラミングの基礎 ・利用法
7	Gaussian 入門 対面	9 月 25 日(木) 13:00-16:30	15	岸本 (理学研究科)	・Gaussian の基本的な使い方
8	はじめての Fortran ハイブリッド	9 月 30 日(火) 13:00-16:00	15	河合 (サイバーサイエンスセンター)	・ Fortran の入門編
9	SX-Aurora TSUBASA の性能分析・高速化 ハイブリッド	10 月 7 日(火) 13:00-16:00	15	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
10	AOBA-B における性能分析と最適化 オンライン	10 月中旬 13:00-16:00	15	担当者 (NEC)	（調整中）
11	NLCPy の利用法 オンライン	11 月頃 13:00-14:15	15	担当者 (NEC)	・NLCPy の基本的な使い方
12	スーパーコンピュータ AOBA を使った高速な統計的機械学習 未定	11 月頃	15	未定	・機械学習の概要説明 ・Frovedis の紹介と AOBA での Frovedis の使い方

［報 告］

2024 年度 サイバーサイエンスセンター講習会実施報告

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	講師	内容
1	はじめての Linux オンライン	6 月 28 日(金) 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 オンライン	7 月 5 日(金) 13:00-15:00	10 (10)	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	7 月 10 日(水) 15:00-17:00	4 (4)	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	ネットワークとセキュリティ入門 オンライン	8 月 2 日(金) 13:30-15:30	22 (22)	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
5	はじめての Linux オンライン	9 月 26 日(木) 13:00-15:00	3 (3)	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
6	Gaussian 入門 対面	9 月 30 日(月) 13:00-16:30	中止	岸本 (理学研究科)	・Gaussian の基本的な使い方
7	並列プログラミング入門Ⅰ (OpenMP) オンライン	10 月 3 日(木) 15:00-17:00	5 (5)	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
8	並列プログラミング入門Ⅱ (MPI) オンライン	10 月 7 日(月) 13:00-16:00	5 (5)	下村 (NEC)	・MPI による並列プログラミングの基礎 ・利用法
9	AOBA-B の最適化 オンライン	10 月 16 日(水) 13:00-16:00	6 (6)	担当者 (NEC)	・AMD コンパイラとプロファイリングツールの紹介
10	MATLAB 入門 対面	10 月 18 日(金) 13:00-16:30	中止	陳 (秋田県立大学)	・MATLAB の基本的な使い方
11	はじめての Fortran ハイブリッド	10 月 28 日(月) 13:00-16:00	4 (2)	江川 (サイバーサイエンスセンター)	・Fortran の入門編
12	SX-Aurora TSUBASA の性能分析・高速化 ハイブリッド	11 月 12 日(火) 13:00-16:00	2 (1)	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
13	NLCPy の利用法 オンライン	11 月 14 日(木) 13:00-14:15	4 (4)	担当者 (NEC)	・NLCPy の概要説明 ・AOBA での利用方法
14	スーパーコンピュータ AOBA を使った高速な統計的機械学習 オンライン	11 月 22 日(金) 13:00-15:00	5 (5)	担当者 (NEC)	・機械学習の概要説明 ・Frovedis の紹介と AOBA での Frovedis の使い方
参加者合計			73 (70)		

［報 告］

後藤英昭准教授らの取組が
「第 7 回日本オープンイノベーション大賞 科学技術政策担当大臣賞」を受賞

当センターの後藤英昭准教授（ネットワーク研究部）らの取組「デジタル社会を支える安全な次世代無線 LAN ローミング基盤の国際共同開発と事業創出・社会実装」が、「第 7 回日本オープンイノベーション大賞 科学技術政策担当大臣賞」を受賞しました。公衆無線 LAN の安全性・利便性向上に向けた、オープンな開発コミュニティの主導と取組が、高く評価されたものです。

【第 7 回日本オープンイノベーション大賞について】
https://www8.cao.go.jp/cstp/openinnovation/prize/2024.html

[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/

コンパイラのバージョンアップについて

2025 年 4 月 1 日に AOBA-A・AOBA-B、2025 年 4 月 7 日に AOBA-S 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	5.2.1	5.3.0	マニュアル&リリースノート
	C/C++ Compiler	5.2.1	5.3.0	
	MPI※1	3.7.0	3.8.0	
	Intel Compiler (VH 用プログラム) ※2	oneAPI 2024.2.0	oneAPI 2025.0.1	oneAPI マニュアル関連
AOBA-B	Intel Compiler ※2			

※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 2025.0.1 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。

Mathematica

- バージョン：14.2
- バージョンアップ日：2025 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）

2025 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。2025 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、以下の 7 件が採択されましたのでお知らせします。

[A] 若手・女性研究者支援課題

No.	申請者	所属	研究課題
A-1	大野　詩歩	東北医科薬科大学 分子生体膜研究所	分子動力学計算による抗 iPS 細胞抗体と糖鎖リガンドの相互作用様式の同定
A-2	小川　秦一郎	大阪公立大学 大学院工学研究科	直交格子積み上げ法を用いた航空宇宙用エンジン内部流れ場の大規模数値解析手法の開発
A-3	笹本　大空	東北大学 大学院薬学研究科	密度汎関数理論に基づいた有機超塩基が示す特異な反応性の理論的考察
A-4	松川　嘉也	東北大学 大学院工学研究科	液滴の帯電・導電挙動に関する直接数値シミュレーション

[B] 萌芽型課題

No.	申請者	所属	研究課題
B-1	松岡　浩	技術士事務所 AI コンピューティングラボ	リカレント型ビット演算による流体・構造体統一解析手法の開発

[C] 一般課題

No.	申請者	所属	研究課題
C-1	山崎　剛	東北大学 大学院理学研究科	日本領域長期再解析(RRJ-Conv.)の長期データ整備と関連する研究開発
C-2	立川　智章	東京理科大学 工学部	データ駆動型手法と高忠実流体シミュレーションの融合による動的流れ制御とその根拠可視化

(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)

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、cc-edu-prog[at]grp.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】
スーパーコンピューティングサポートユニット（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】
スーパーコンピューティングサポートユニット（cc-uketuke[at]grp.tohoku.ac.jp）

（情報部デジタルサービス支援課）

2025 年度利用負担金について

2025 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）

執筆要項

— 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。詳細はスーパーコンピュータサポートユニット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スーパーコンピューティングサポートユニット
e-mail　cc-uketuke@grp.tohoku.ac.jp
TEL　　022-795-3406

スタッフ便り

3/1 付けで名古屋大学 情報基盤センターよりこちらに着任いたしました、河合直聡と申します。
関東以北に住むのは初めてで、スキーが趣味ということもあり、楽しみに引っ越してきたら、新しい家では電気を通せておらず、水道を開けたらエコキュートから水が噴水のように噴き出し、慌てて不動産会社に連絡を取ろうとしたら携帯が壊れているなど、滑り出しで派手に転倒いたしました。ということで、これ以上こけないように地に足をしっかりつけて研究、教育、業務に携わっていきたいと思います。
どうぞよろしくお願いいたします。🏠(M. K)

先日、初めていちご狩りに行きました。山元町にあるいちご農園さんに 10 時半の開始で予約をしていましたが、到着したときには駐車場がほぼ満車の状態で、私がお邪魔したいちご農園さんは 9 時から営業していたようです。駐車場に停めてある車は県外ナンバーも多くあり、想像以上に人気があることを知りました。
私が参加したのは 30 分の食べ放題で、10m～15mほどのレーンが両サイドに 2 レーンあり、2 レーン分を参加グループごと（2 人ずつ）に割り当てられていました。食べ始めるまでは、たくさん食べようと意気込んでいましたが、開始 10 分ほどでだいぶ満足してしまいました。(結構な量を食べたつもりなのですが、時間配分を誤ったようです・・・)
残りの時間は、農園のスタッフさんとお話ししていましたが、スタッフさんのお話しによると、いちご狩りはハウスに人が入っていない朝一の時間帯のほうがいちごの状態がいいとのことです。山元町にはいちご農園がたくさんありますが、土日祝日のタイミングによっては、どのいちご農園さんも予約でいっぱいになるそうです。
いちご狩りに行くことを検討している方は、朝一での参加がおすすめですよ。🍓(E. A)

【サイバーサイエンスセンター・情報部デジタルサービス支援課スタッフ　着任・採用のお知らせ】

2025. 3. 1 着任
河合　直聡　スーパーコンピューティング研究部　助教(名古屋大学情報基盤センター特任助教から)

2025. 4. 1 着任
横川三津夫　高性能計算技術開発 (NEC) 共同研究部門　特任教授 (研究) (神戸大学教授から)

2025. 4. 1 採用
藤村　瑠菜　スーパーコンピューティングサポートユニット　情報一般職員
一井　真衣　総務係　事務補佐員

2025 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
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
共有	sxs	1	1VE	72 時間/720 時間	96GB×VE 数
		1～2048	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-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.58 No.2 2025−4