1. 調査研究の概要

1.1 調査研究の背景と目的

次世代の計算基盤には、従来のサイエンスにとどまらず、広範なSDGs・Society 5.0の実現に向けた課題解決のためのプラットフォームとしての役割が一層求められる。特に、今後の科学では、いわゆる「研究DX」により新たなイノベーションを起こすべく非常に高度なデジタルツインを中心としたものとなり、そのためには現象をモデル化することによる演繹的シミュレーションと、学習・AIに基づく現象の帰納的シミュレーション、さらには大規模なデータを収集・処理基盤と、データの両方のシミュレーションとの同化、などが密に連携する計算基盤の構築、及びその開発・運用が鍵となる。このような次世代計算基盤では、大規模高速な計算・データ処理が行えるスーパーコンピュータがその中心的な役割を担う。特に、比較的狭いシミュレーション領域を対象とした過去のものや、AIや特定領域の計算に特化したマシンと異なり、デジタルツインの基盤としてのスーパーコンピュータは、広範な計算手法・シミュレーション技法や大規模データを駆使しつつ、それらが密に連携しながら全体のワークフローの実行が可能である必要があり、単に特定のベンチやアプリで高い性能を達成するのみならず、幅広いアプリケーション分野で高効率を達成できる計算基盤として特質が要となる。

しかしながら、その実現は非常に困難を極めると予想される。一般に、高い性能と広い適用可能性は相反する要求でもあり、現在理化学研究所計算科学研究センターで運用されているスーパーコンピュータ富岳の開発では、ハードウェアとアプリケーションのコデザインをHPCの産学オールジャパン体制で推し進めることで、それを実現可能なものとした。一方、次世代計算基盤では、ムーアの法則の終焉が近づきつつあるなか、半導体プロセス技術の進歩による演算あたりの大幅な電力改善が見込めず、高い性能をあまねく達成することはさらに難しくなると予測されている。実際、計算機科学コミュニティが調査・執筆したNGACI白書でも、2028年度に実現可能なシステムでは、許容電力の大幅増を仮定しても、富岳の3.3倍から10倍程度の性能向上にとどまると予想されている。また、近年のソフトウェア開発における複雑性の増大も、高い性能と広い適用可能性への大きな足かせになる。従来型の浮動小数点演算ピーク性能重視のシステム設計では、一部のベンチマークでは一見華々しい性能を見せつつも、実際のアプリケーションとの性能乖離は絶望的となり、今後立ち行かなくなることは明らかであり、またソフトウェアのエコシステムの重要さやコストを軽視した計算基盤では、開発しても使うことのできないシステムになるであり、研究DXの基盤としては意味のないものとなる。

性能面に関しては、様々な文献や我々のベンチマークや性能モデルの多岐にわたる研究調査でも、大多数のアプリケーションはメモリやネットワークバンド幅など、基本的にデータ移動の性能によって律速される特徴を持つことがわかっている。次世代計算基盤では、FLOPSで語られるピーク性能重視から脱却し、データ移動、すなわちByteにより表現される指標をアーキテクチャとアルゴリズムの点から最適化・高効率化していくことが重要となる。本提案では、上記理念の達成に向け、アーキテクチャ設計の基本理念として演算精度も考慮しながら必要な計算性能は確保しつつ、電力制約の下でデータ移動を高度化・効率化する"FLOPS to Byte"指向のシステム構築を、アーキテクチャ開発からアルゴリズム設計、アプリケーション技術に至るまで実践し、実効的な性能を向上させるための次世代計算基盤を調査研究する。

アーキテクチャの調査研究では、電力制約の下でデータ移動を高度化・効率するデバイス技術やアーキテクチャ技術を中心に調査研究を実施する。また、上記の近年までの大規模スーパーコンピュータでは、レイテンシよりも演算とデータ転送スループットを重視し、かつ並列度を上げた際の実効効率低下を緩和させるために、弱スケーリング思想に基づく設計が重視されてきた。一方、機械学習における推論処理や創薬系のアプリケーションでは同一サイズの問題を高速に解く強スケーリングが求められ、半導体技術によるDRAM容量の増加傾向も鈍化しつつある。これら背景を踏まえ、アーキテクチャ検討項目として、特に3次元積層メモリ技術の活用、データフロー的観点も踏まえた強スケーリング向けの要素技術、チップ間直接光通信技術を特に意識しながら、関連するアーキテクチャ技術を国内外複数ベンダと共に調査研究を実施する。

システムソフトウェア・ライブラリの調査研究に関しては、これまでのフラッグシップや第二階層システムのソフトウェアの開

発工数や利用状況、そして既に構築されているエコシステムとの親和性も踏まえ、ソフトウェア資産として何をベースに、国内でどこまでを開発すべきか、もしくは国際協調が必要なソフトウェアに優先度をつけて明らかにしつつ、今後のロードマップを策定することが必要である。また、次世代計算基盤を産業界も含む幅広い応用分野での活用を促すためにも、従来のシステム・ソフトウェアだけでなくデータ利活用の促進、機械学習技術と第一原理シミュレーション、さらには大規模リアルタイムデータ処理の高度な融合、従来の共用HPCシステムとは次元の異なる高セキュリティの担保、などを主要検討項目と見据えた今後の日本の次世代計算基盤として行うべきソフトウェア開発について調査研究を行う。

アプリケーションの調査研究に関しては、従来のサイエンスをさらに進化・深化させるのみにとどまらず、社会科学やSociety 5.0といった新しい応用分野への展開も見据え、アーキテクチャとアルゴリズムとアプリケーションの三者が連携するコデザインに基づく次世代計算基盤構築に向けた調査研究を実施する。特に、複数アーキテクチャを統一的に評価するための広範なベンチマークセットを構築し、それを利用したアーキテクチャ評価結果を踏まえてアルゴリズムやパラメータの改善を検討するほか、それをベンチマークセットへと更新し、性能モデルを構築した上で、探索的な”What if” する評価を行う。このサイクルを回した新たなコデザインにより、各アプリケーションで高い実効性能を得るためのアーキテクチャとアルゴリズムを探求する。これにより、一部のベンチマークやアプリケーションのみに特化した性能重視の罠に陥ることなく、一方、単にあらゆるアプリに対し八方美人であることは避け、今後アーキテクチャの進化に伴って、どのようなアルゴリズムのクラスが大幅な進化が見込まれるか、という指標も抽出し、今後の発展につなげる。

フラッグシップシステムの開発にあたっては、日本の半導体戦略とも整合していくことが重要であり、特に今後の日本が持つべき半導体技術の強みを活用しつつさらにそれを伸ばすための戦略や、開発されたシステムの幅広い産業展開を見据えた調査研究へのフィードバックも、運営委員会や他調査研究チームと連携しつつ実施する。これらを総合し、ポスト「富岳」時代の次世代計算基盤の具体的な性能や機能等について検討を行うことが目的である。

1.2 調査研究の体制

上記の調査研究を実施するために、国内外の主用なHPC関連のベンダと、HPCI第二階層を構成する国内のスーパーコンピューティングセンタ、また様々なアプリケーション分野をリードする国立研究所の研究者と共に、以下の体制で調査研究を実施する。

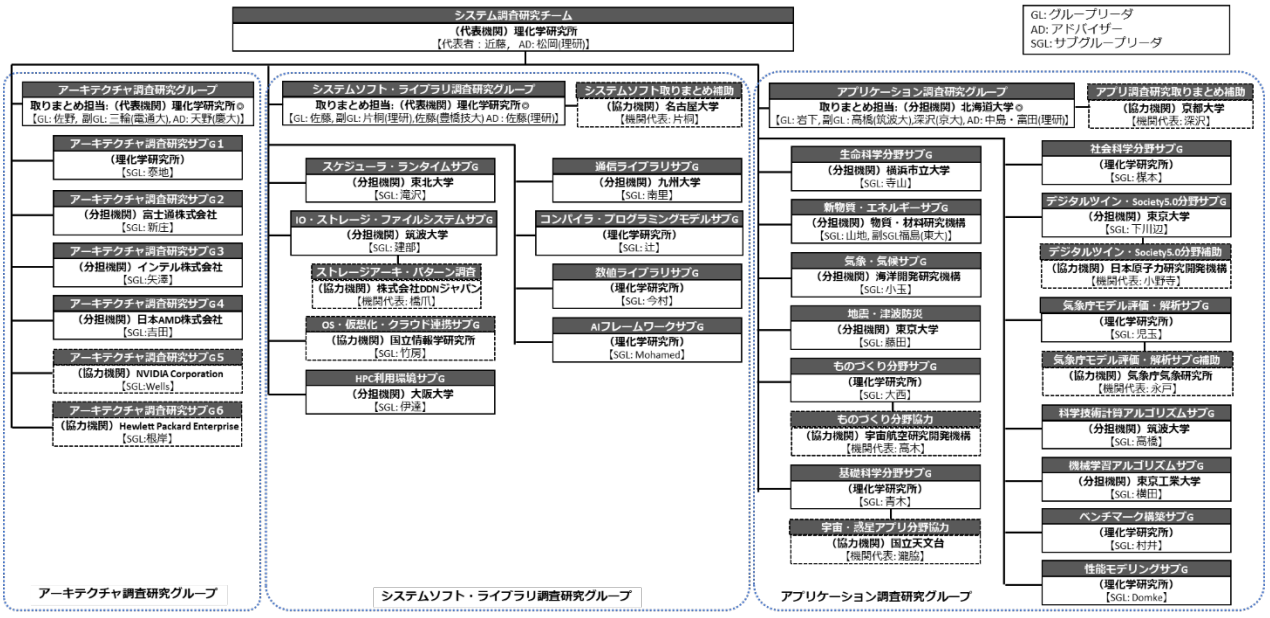


図 1.2.1 調査研究の体制

2. アーキテクチャ研究グループ

2.1 調査研究の概要および方針

2.1.1 目的と方法

アーキテクチャ研究グループでは、科学技術研究分野、産業分野、および社会からの利用ニーズに対して有望なシステムの選択肢の探索およびその実現可能性の提示を目的とし、ムーア則の終焉や半導体技術およびパッケージング技術等の発展を見据えつつ、システム全体やその構成要素について考え得る技術的可能性に関する調査検討を行う。そのために、例えば、1. 汎用プロセッサ（CPU）、2. DRAM、3. SRAM（キャッシュメモリ等）やその3次元積層技術、4. アクセラレータ、5. 相互接続網、6. ストレージ、7. CMOS技術やパッケージング技術（3次元積層、チップレット等）、8. シリコンフォトニクス、9. 信頼性・可用性・保守性、などについて、技術動向と共に次世代計算基盤システム開発に適した将来技術を調査・探索し、その有効性および実現可能性を評価する。

ハードウェアシステムの開発および製造・設置に関しても具体的な検討を行うために、アカデミアに加え、関連分野において長年に渡り技術開発とシステム販売に携わってきたベンダを分担機関あるいは協力機関とし、上記の調査検討を実施する。特に、昨今の半導体業界やITC業界では国境を跨ぐ世界的なサプライチェーンや技術連携が不可欠であり、次世代計算基盤開発においても日本単独で全ての技術を網羅することは難しいと予想されることから、ベンダとして国内外の企業に参画頂き、特に日本が独自に保有し発展させるべき技術と国際的に連携すべき技術の候補を明らかにしつつ、開発すべきシステムとそのアーキテクチャの選択肢を提示することを目指す。半導体技術等の原理的限界と、技術的に突破できる見込みのあるそれ以外の課題を精査し、次世代計算基盤に求められる要件に対して優先順位を設定した上で解決策の実現可能性を検討する。

以上の調査研究を実施するために、グループとりまとめに加え以下のサブグループを設置した。

A0. アーキテクチャグループとりまとめ　　担当機関：理化学研究所（代表機関）
A1. アーキテクチャ調査研究サブグループ1　　担当機関：理化学研究所（代表機関）
A2. アーキテクチャ調査研究サブグループ2　　担当機関：富士通株式会社（分担機関）
A3. アーキテクチャ調査研究サブグループ3　　担当機関：インテル株式会社（分担機関）
A4. アーキテクチャ調査研究サブグループ4　　担当機関：日本ＡＭＤ株式会社（分担機関）
A5. アーキテクチャ調査研究サブグループ5　　担当機関：NVIDIA Corporation（協力機関）
A6. アーキテクチャ調査研究サブグループ6　　担当機関：Hewlett Packard Enterprise（協力機関）

「A0. アーキテクチャグループとりまとめ」では、以下の項目について調査研究を実施した。詳細は、それぞれの節に記載する。

- 半導体技術・パッケージング技術・アーキテクチャの技術動向の調査（2.1.2 節）
- ベンチマークプログラムの特徴解析によるワークロード分析（2.1.3 節）
- 次世代計算基盤への要件や評価指標の整理、システムとしてあるべき姿の検討（2.1.4 節）
- サブグループ調査研究の方向付けと推進、内容のレビューやフィードバック（2.1.4 節）

アーキテクチャ調査研究サブグループ1～6では、マイクロアーキテクチャ、ノードアーキテクチャ、システムアーキテクチャ、および関連するシステム・ソフトウェアやベンチマークによるアプリケーションの性能推定等の個別に設定する調査研究対象について、技術的な調査検討を行った。詳細は、2.2～2.7節について述べる。

2.1.2 技術動向調査

2.1.2.1 技術動向調査の概要

スーパーコンピュータ、特にフラッグシップ機は、高い演算性能と電力性能を実現するために最先端のハードウェア技術を駆使してシステム構築が行われる。そのため、次世代計算基盤のアーキテクチャ検討においては、システムを構成する様々なハードウェアの技術動向を十分理解した上で、2028年頃までに実用化可能な技術を見極める必要がある。

そこで本研究グループでは、次世代計算基盤のアーキテクチャ検討に先立ち、種々のハードウェア技術について動向調査を行った。本研究グループが動向調査の対象とした技術は、具体的には、半導体、パッケージング、シリコンフォトニクス、汎用プロセッサ、メモリ、アクセラレータ、ノードアーキテクチャ、CPU接続、ネットワークである。以下では各技術の動向調査結果を報告する。

2.1.2.2 半導体・パッケージング・シリコンフォトニクス

半導体の微細化は2030年でも継続する見通しである。IRDSテクノロジロードマップによると、1 nm eq プロセスが2031年に実現される見通しであり、5 nmプロセスから1 nmの微細化で面積あたり4倍以上の実装密度を実現する。トランジスタ構造は3 nm近辺を境に、finFETからLGAAに変化する。ウェハコストは微細化とともに上昇する傾向があるが、ダイ面積も微細化とともに縮小するため、同一機能を有するダイの製造コストは微細化により低下する。ウェハ上の欠陥密度はプロセスによらず概ね一定の傾向（TSMC N7プロセスで0.09 欠陥/cm^2）があるため、同一機能を持つダイを製造する場合、チップ面積に比例して歩留まりは改善する傾向にある。以上より、微細化による性能改善・価格低下・歩留まり改善は依然として進展する。

一方で、計算重視の計算基盤を持続的に成長させる上で、微細化の鈍化が今後の大きな懸念事項である。さらに前述の通り、ウェハ上の欠陥密度はプロセスによらず概ね一定の傾向があるため、ダイ面積を拡大して並列ユニット数を増やすと歩留まりが低下し、逆に製造コストの増大に繋がる問題点がある。このため、ダイを複数枚に分割して、2次元方向に接続するチップレット技術や3次元集積技術が今後の技術トレンドとして研究されている。NVIDIA社GPUのチップレット技術トレンドを見ると、チップを接続するインターポーザのサイズが3年ほどでおよそ倍増のペースで成長しており、演算ユニットに搭載するトランジスタ数は直近5年で20倍、メモリ搭載数が8倍に進化している。複数のチップレットを相互接続するための規格としてUniversal Chiplet Interconnect Express（UCIe）が2022年頃から急速に策定されはじめており、機能をわけたチップレットの相互接続を目指して今後の持続的発展が注目されている。従来のチップ大面積化による機能改善と代わって、省面積チップレットの3次元統合が設計コストを下げながら性能改善を実現する有力手段として考えられる。

3次元実装方式として、積層されたチップの電気接続をThrough Silicon Via (TSV)により行うことが現在主流であり、メモリや演算ユニットの実装密度を高める。メモリチップを3次元方向にTSV接続したHBM (High Bandwidth Memory)は近年では10層以上積層し、その容量は10 GBを超えている。この3次元積層されたチップを水平方向（2次元方向）に密に並べるために、TSVやインターポーザを排除したEmbedded Multi-die Interconnect Bridge (EMIB) がIntel社により開発されており、2次元方向への高密度実装の開発も活発に行われている。

データセンタなど、データ転送がボトルネックとなるアプリケーションに対して、スイッチング方式の最適化などによりスループット帯域の改善は行われている一方で、サーバやラック間のデータ転送を行うモジュールの速度が追いついていない。1つのブレークスルーとしてシリコンフォトニクス技術が重点的に研究されている。電気配線を用いた通信では配線の寄生成分によりバンド幅が狭くなり、電力のロスが発生する一方で、光通信はこのようなバンド幅の低下や電力ロスが深刻ではないためである（現行の光モジュールの性能については、2.1.2.5を参照）。光通信モジュールと、ASIC間は現状では電極を介してプラグ接続されているため、集積度やバンド幅、電力効

率が律速される問題点がある。先述したチップレットの相互接続技術の急速な発展を背景に、Co-packaged Optics (CPO)が重点的に開発されている。例えば光変調器を搭載したプロセスで製造し、ドライバや制御回路をCMOSチップレットで混載設計することで、基板やパッケージ間を繋ぐ配線や減衰信号補償用のDSPが不要になる。しかし、依然としてデータセンタに必要なスループット帯域に追いつく性能には至っておらず、1つの問題点として集積度がCMOS回路より悪い点が挙げられる。チップレット積層技術や光素子そのものの高集積化が重要課題である。

2.1.2.3　汎用プロセッサ・メモリ

- **汎用プロセッサ**：以下では汎用プロセッサの技術動向をまとめる。

  HPC用途では主にIntel、 AMD、 IBM、 富士通の各ベンダが汎用プロセッサを提供しており、世界トップレベルの性能を有するスーパーコンピュータは上記のいずれかのベンダのプロセッサを採用することが多い。HPC向けに設計された汎用プロセッサでは多数のコアと各コアに搭載したSIMD演算ユニットによってピーク演算性能を稼ぐアーキテクチャを採用する傾向にあり、現在は数十個のコアとコアあたり1～2個の512ビットSIMD演算ユニットを搭載したものが主流となっている。また、Intel第4世代Xeonスケーラブルプロセッサ やIBM Power10等の一部のプロセッサでは、AI処理で多用される行列演算を加速するために行列演算ユニットの搭載を開始している。上記の傾向は今後も続くと予想されることから、ソケットあたりのコア数、SIMD演算ユニットの幅と数、行列演算ユニットの幅と数が2028年頃にどの程度まで増大するかは注意深く見守る必要がある。なお、2023年4月現在、HPC向け汎用プロセッサのピーク演算性能は最大で5.38TFLOPS（AMD EPYC 9654Pにおける倍精度浮動小数点演算性能）に達しており、TDPは350W前後、電力性能は十数GFLOPS/W程度である。メインメモリに関してはこれまでDDRが主体であったが、近年は富士通A64FXやIntel Xeon-MaxなどのHBMをサポートする汎用プロセッサも登場し始めている。HBMをサポートする汎用プロセッサはバンド幅律速のアプリケーションに対してGPUとほとんど変わらない性能を示すと期待されることから、このようなプロセッサの技術動向は今後注視していく必要がある。また、拡張インターフェースとしてPCIe 5.0を数十から百数十レーン搭載したものが主流となっている（IBM Power9で採用されたNVLinkはPower10では廃止されている）。

  半導体製造技術とアーキテクチャの進歩によって汎用プロセッサの性能は大きく向上しているものの、単純にピーク演算性能や電力性能で比較すると、汎用プロセッサはGPUを始めとするアクセラレータとは依然として10倍以上の開きがあるのが現実である。汎用プロセッサがアクセラレータよりも電力性能で劣るのは、命令をアウトオブオーダ実行するための様々なユニットが汎用プロセッサには必要であり、演算以外の部分で消費される電力が大きいためである。これはプロセッサが汎用であり続けるための本質的な性質であり、汎用プロセッサとアクセラレータの電力性能比は2028年頃においても現在と同程度であることが次世代先端的計算基盤に関する白書でも予想されている[1]。また、近年のスーパーコンピュータ開発においてはシステムに課された電力バジェットが大きな制約となっており、この制約は今後厳しくなることはあっても緩くなる可能性は低い。したがって、2028年頃に高いピーク演算性能を有するシステムを実現する上では、何らかのアクセラレータの採用は避けて通ることができないものと思われる。その一方で、アクセラレータによって加速可能なアプリケーションは一般に一部のドメインに限られることから、アクセラレータを採用する場合は当該アクセラレータが加速可能なアプリケーションのドメインを慎重に検討するとともに、当該アクセラレータでは加速できないアプリケーションの加速方法についても考える必要がある。

- **メモリ**：ここでは、主にオンチップおよびオフチップのメモリ技術の動向調査についてまとめる。

  オンチップメモリとして主要なメモリ素子としては SRAM が用いられている。これまでは、SRAMは基本的にプロセスのスケーリングに伴って微細化され、高集積化と低消費電力化を実現してきた。しかし、10nm 以

降のテクノロジではロジックの微細化は鈍化しつつも進んでいるが、配線に関しては微細化されると抵抗が増加することもあり、ロジックとの乖離が広がっている。キャッシュなどに用いられる大容量 SRAM は配線領域が支配的となるため、大容量化が難しくなってきている。また、電圧についても安定動作のためには現在の電圧が限界となり、これ以上の低電力化は難しくなってきている。一方で、ハイブリッドボンディングやTSV（スルーシリコンビア）による3次元実装技術により、SRAM 層をロジック層の上に搭載することにより、高バンド幅や大容量化を実現する試みも一部で実用化されており、3 次元実装技術の進展がオンチップメモリでも重要となってきている。また、3次元実装技術では、DRAM をオンチップに搭載することも可能となるため、大容量化の方向としては検討項目となる。ただし、アクセス速度は SRAM に比べると遅いため、階層化は必須となる。また、DRAM ではリフレッシュが必要となるなど課題も多いため、これらを解決する不揮発型メモリの開発も行われており、実用化に向けた動向について注意すべきである。

オフチップメモリとして最も使われているのは DRAM で構成されるデバイスである。DRAM においては、第 5 世代の規格である DDR5 が2021年末から発売が開始され、最新の CPU では利用され始めている。DDR5 は、チップモジュールあたり 16Gbit の容量を有し、1.1V 供給電圧の下で、現在出荷されている DDR5-4800 では 38.4GB/s であるが規格的には 51.2GB/s までを視野に入れている。次世代規格として DDR6 が出てくるのは 2027 年頃と見られ、その動向に注意が必要である。一方、より高いメモリバンド幅を要求する HPC 向けのプロセッサ（富士通 A64FX や Intel Xeon-Max など）やアクセラレータ（NVIDIA H100 や NEC SX-Aurora TSUBASA など）には TSV とシリコンインターポーザを用いた HBM が用いられている。HBM は DDR と比べて高いバンド幅と電力効率を有し、最新の HBM2E では 410GB/s のメモリバンド幅を実現している。次世代の HBM3 の仕様書も 2022 年初めに JES238 として公開され、ピンあたりのデータ転送速度を 2 倍に相当する 6.4Gbps に拡張し、デバイス当たり 819GB/s とした。TSV スタックも最大 12 とし、将来的には 16 への拡張性も確保している。層あたりの容量密度も8Gb から 32Gb とし、デバイス当たり 64GB までサポートできる。ただし、初期の製品は 16Gb となる見込みである。一方で、これらの二つのメモリデバイスには、容量、メモリバンド幅、コストのトレードオフがある。このため今後は、これらのトレードオフを考慮しながら、DDR と HBM が引き続き高性能計算システムのメモリ素子として利用されていくことが予想される。さらに、新しい不揮発メモリデバイスについても、開発が進められており、電力・コスト・容量などの観点から動向を注目する必要がある。

[1] NGACI、White Paper on Next-Generation Advanced Computing Infrastructure、ver. 1.0.0 (2020)

2.1.2.4　アクセラレータ・ノードアーキテクチャ・CPU 接続

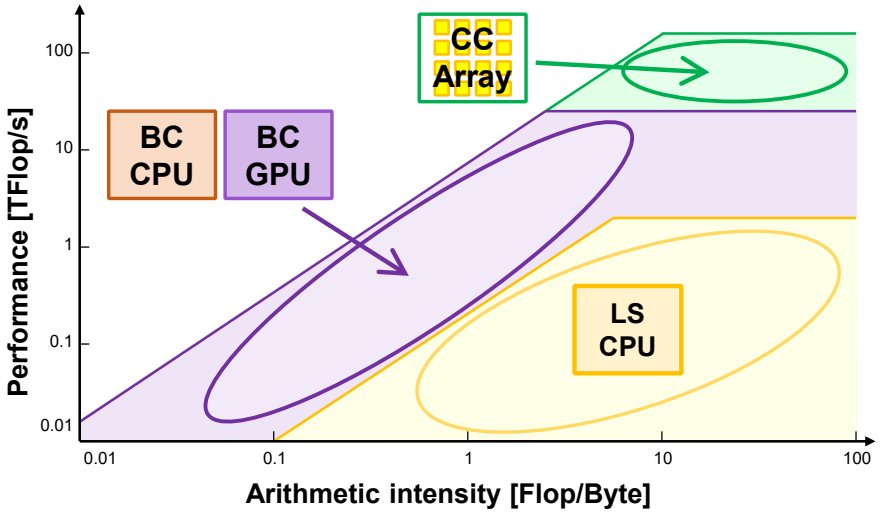


図 2.1.1 Roofline モデルとノードを構成する基本アーキテクチャ

次世代先端的計算基盤に関する白書（NGACI，https://sites.google.com/view/ngaci/home）にもまとめられているように、高性能計算機を構成可能な基本アーキテクチャ要素としては、

- Latency Sensitive CPU [LS]
- Bandwidth Centric CPU または GPU [BC]
- Compute Centric Accelerator [CC]

の3つが考えられる。図 2.1.1は、Rooflineモデル上においてこれらの基本アーキテクチャの達成可能性能領域を示したものである。横軸は演算強度[Flop/Byte]で、縦軸は性能[TFlop/s]を示す。LSはデータセンタ等で用いられるようなハイエンドCPUのことであり、スレッド処理の遅延を短縮して様々なワークロードに対する性能を高めるような汎用プロセッサである。その特性、および大容量の主メモリを搭載するために、DDRメモリが採用されていることが多い。また、遅延を削減する様々な機構のためにピーク性能や電力性能比はその他の基本アーキテクチャよりも低い範囲となる。

BCは、HBMメモリを搭載した計算処理のスループットを指向したハイエンドCPUやGPUであり、例えば、富岳の富士通 A64FX CPU、インテル Sapphire Rapids CPU、NVIDIA H100 GPUがそれに該当する。個々のスレッドの遅延を低減する代わりに処理全体のスループットを向上させることに主眼を置いたアーキテクチャであり、そのために、容量を多少犠牲にしながらもHBMなどの広帯域メモリを採用する。スループット指向により遅延の削減機構に多くの資源を割かずに済むため、電力性能比はLSよりも高く、結果として高いピーク性能を有する。CCはメモリ帯域よりも演算そのものの性能を高めるため多数の演算を集積したアーキテクチャであり、その目的のために、特定の計算処理や対象とするアルゴリズムドメインに特化したハードウェア構成を有することが多い。例えば、Google TPUや、NVIDIA V100 GPUから搭載され始めたTensor コア、あるいはSystolic arrayやCoarse grained reconfigurable array (CGRA)といった、処理の空間的並列性や時間的並列性（パイプライン処理）を最大限に引き出すようなアレイ型のアクセラレータがCCに該当する。特化型のハードウェア構成により、高いピーク性能や電力性能比を持つことが可能である。

図 2.1.1に示す通り、メモリ帯域に性能が制約されるメモリバウンドから演算のみに性能が制約されるコンピュートバウンドまでの広範なアプリケーションに対応するには、LS CPUのみの構成では不十分であり、BC型アーキテクチャを導入しながらも必要に応じて不足するコンピュートバウンドの演算性能を補うCC型アーキテクチャを組

み合わせる構成が有望であると考えられる。ノードアーキテクチャの設計では、組み合わせる基本アーキテクチャの種類と、アーキテクチャごとの性能および電力のバランスが重要な指針となる。例えば、富岳A64FX CPUのような広帯域メモリを搭載したBC CPUにGPUやあるいはアレイ型のCCアクセラレータを組み合わせるといった構成が検討の対象となり得る。これらの背景を踏まえ、BC型やCC型アクセラレータ、それらを搭載したノードアーキテクチャやCPU接続の技術動向を以下にまとめる。

- アクセラレータ

スーパーコンピュータに対する要求性能と利用可能な電力容量の制限、昨今の脱炭素化へ向けた動向等からスーパーコンピュータの電力効率の向上は喫緊の課題であり、その解としてGPUをはじめとした演算加速装置 (アクセラレータ) の活用が現在の主流となりつつある。その証左として、図 2.1.2に示す2022年11月のGreen500の上位10システムは全てアクセラレータを搭載しており、それらシステムの殆どがNVIDIAもしくはAMD製のHPC向けGPUを採用したスーパーコンピュータであるのが現状である。

The GREEN 500

MOST ENERGY EFFICIENT ARCHITECTURES

TOP 500

Computer		Interconnect	Accelerator	Rmax/ Power
Henri, Lenovo ThinkSystem SR670 V2	Intel Xeon 32C 2.8GHz	Infiniband HDR	NVIDIA H100	*65.1
Frontier TDS, HPE Cray EX235a	AMD EPYC 64C 2.0GHz	Slingshot-11	AMD Instinct MI250X	*62.7
Adastra, HPE Cray EX235a	AMD EPYC 64C 2.0GHz	Slingshot-11	AMD Instinct MI250X	*58.0
Setonix-GPU, HPE Cray EX235a	AMD EPYC 64C 2.0GHz	Slingshot-11	AMD Instinct MI250X	57.0
Dardel-GPU, HPE Cray EX235a	AMD EPYC 64C 2.0GHz	Slingshot-11	AMD Instinct MI250X	56.5
Frontier, HPE Cray EX235a	AMD EPYC 64C 2.0GHz	Slingshot-11	AMD Instinct MI250X	52.2
LUMI, HPE Cray EX235a	AMD EPYC 64C 2.0GHz	Slingshot-11	AMD Instinct MI250X	51.4
Atos THX.A.B, BullSequana XH2000	Xeon 32C 2.4GHz	NVIDIA HDR100	NVIDIA A100	*41.4
MN-3, Preferred Network MN-Core Server	Xeon 24C 2.4GHz	RoCEv2/MN-Core DirectConnect	MN-Core	40.9
Champollion, Apollo 6500	AMD EPYC 64C 2.45GHz	Mellanox HDR	NVIDIA A100	38.6

[Gflops/Watt]

図 2.1.2 SC22 (Green500 BoF) 発表資料

1位を獲得したのは、ニューヨークのフラットアイアン研究所にLenovoによって導入された「アンリ（Henri）」であり、このシステムが搭載するH100 GPU (アーキテクチャ名：Hopper) はNVIDIAが現在 (2023年4月) 提供している最新型GPUである。本GPUの製造プロセスルールは4nmであるため、7nmプロセスを採用した前世代のNVIDIA A100と比較して演算コア数は22%増加、動作周波数は約1.3倍向上している。その恩恵によって、IEEE FP64およびFP32のピーク演算性能はそれぞれ30 TFLOPS、60 TFLOPS (SXM5)に達しており、これらはA100の約3.1倍の性能である。また、これらの性能をTDPに基づいた電力対性能に換算するとH100 (TDP：700W)はA100 (TDP：400W)の約1.8倍の電力効率を達成している。さらに、今日の深層学習やAIワークロードの高速化の需要の高まりから、NVIDIAはH100の2世代前のGPUであるV100からTensorコアと呼ばれる行列積和演算に特化した演算回路を新たに搭載している。V100ではFP16精度と、Turing Tensorコアで追加されたINT8、INT4、バイナリ1ビット精度のデータ型をサポートしていたが、A100からはそれらに加えてTF32、BF16、FP64 形式のデータ型がサポートされ、H100のTensorコアによるFP16、TF32、FP64のピーク演算性能はそれぞれ1000、500、60 TFLOPS

にまで達している。なお、これらの性能とTensorコアを使用しない場合の性能差はそれぞれ8.3、8.3、2倍となる。

2位から7位には2022年6月、 11月のTOP500で首位となっているスーパーコンピュータFrontier (米国オークリッジ国立研究所)と同じ構成のシステム群がランクインしている。システムに搭載されたアクセラレータはAMDの提供するHPC向けGPUであり、MI250Xは現時点 (2023年4月) において最上位モデルとなる製品である。製造プロセスルールは6nm、IEEE FP64およびFP32のピーク演算性能はどちらも47.9 TFLOPSに達しており、FP64の理論ピーク性能に限って言えばNVIDIA H100 GPUを上回っている。また、TDPは500W (最大560W) であるため、MI250Xの電力効率は理論的にはNVIDIA A100 GPUの約3.2倍であり、かつMI250Xを搭載したFrontierベースのシステムはGreen500ランキングの大半を占めていることから、その電力効率の良さを実際に示している。そして、AMD MI250X GPUにおいても深層学習やAIワークロードの高速化の需要に対応すべく、行列積専用命令をサポートするCDNA 2アーキテクチャを採用しており、これによるFP64行列積のピーク演算性能は95.7 TFLOPSにまで達する。

NVIDIAやAMD以外に、IntelもHPC用途のGPUを開発 (コードネーム：Ponte Vecchio) しており、米国アルゴンヌ国立研究所に導入予定のスーパーコンピュータAuroraにアクセラレータとして搭載される計画である。GPU以外のBCアクセラレータとしては、NEC SX-Aurora TSUBASAシリーズのようなベクトルプロセッサが存在し、図 2.1.1で触れたCC型アクセラレータとしては、Green500の9位にランクインしたMN-Coreをはじめとした深層学習やAIワークロードの加速を目的とするASICベースのアクセラレータや特定のニーズに合わせた演算加速を実現する回路を柔軟に実装出来るFPGAなどが挙げられる。CC型アクセラレータの中でも特に、Cerabras社はAIに最適化された演算コアとローカルメモリ全てを1枚の大型半導体チップに統合するという挑戦的なアーキテクチャを採用している。Cerabras社はこのアーキテクチャに基づいたチップをWSE - Wafer Scale Engineと呼んでおり、このWSEを搭載したシステムであるCS-1のFP32のピーク演算性能は3.3 PFLOPSにまで達している。CS-1は既に米国アルゴンヌ国立研究所をはじめとする様々な研究機関に導入されており、国内でも、2019年12月に東京エレクトロンデバイス株式会社が Cerebras CS-1の代理店契約を正式に締結し、受注を開始している。そして、WSEの第二世代であるWSE-2を搭載したCS-2も既に稼働しており、CS-1の最大2倍の性能を発揮することが見込まれている。

先述したように、スーパーコンピュータの開発において現時点における最大の問題は電力問題であり、いかにして電力効率を向上させていくかが重要である。そのための手段として最有力と目されているのがGPU等のアクセラレータの活用であり、Green500はその証左となっている。従って、今後においてもヘテロジニアスコンピューティングのためのハードウェアおよびソフトウェアの研究開発はますます重要になることが予想される。

- ノードアーキテクチャ・CPU 接続

2023年現在では、アクセラレータを搭載したノード構成においては、アクセラレータを制御するCPUはx86のISAをサポートしたメニーコアが広く用いられており、これがノードあたり1～2 個搭載されている。ホストメモリとして現在広く用いられているのはDDR規格のDRAMで構成されるデバイスであり、CPU (ソケット) あたりにDDR4メモリモジュールが複数チャネル接続される構成が一般的である。例えば、ソケットあたりに32 GiB DDR4-3200メモリモジュールが8チャネル接続される場合、メモリサイズは256GiBであり、メモリバンド幅は204.8 GB/sとなる。ただし2.1.2.3で述べられている様に、2021年末から第5世代の規格であるDDR5メモリの発売が開始されており、現在筑波大学計算科学研究センターで運用中のスーパーコンピュータPegasusでは、ノードあたり (CPUあたり) に16 GiB DDR5-4400メモリモジュールが8チャネル接続されている。従って、メモリサイズは128GiBであるが、メモリバンド幅は281.6 GB/sとなる。

CPUとアクセラレータ間のインターフェースとして広く用いられているのはPCIeであり、NVIDIAが現在 (2023年4月) 提供している最新型GPUであるH100 GPU (PCIe版)は5.0規格をサポートしている。ま

た、PCIeはノード間接続のためのネットワークカード (例：InfiniBand HCA) やNVMeのインターフェースとしても用いられるため、ノード内のPCIeデバイスはPLXと呼ばれるPCIeスイッチを介して接続される場合が多い。PCIe以外のインターフェースとしては、NVIDIAが提供するNVLinkやAMDが提供するInfinity Fabricがあり、前者はスーパーコンピュータSummit (米国オークリッジ国立研究所)、後者はスーパーコンピュータFrontier (米国オークリッジ国立研究所) で採用されている。これらのテクノロジは同世代のPCIeインターフェースと比較して数倍以上の帯域幅を有する以外にもキャッシュコヒーレンシをサポートすることからGPUプログラミングが簡素化されるという利点があり、アプリケーション開発者の注目を集めている。そして現在、CPUとGPU を1パッケージ化して両者を更に密結合させるアーキテクチャについてNVIDIA、AMD、Intelの各社は注力しており、NVIDIAはGrace-Hopper Superchip、AMDはMI300、IntelはFalcon Shoresのリリースを計画している。

2.1.2.5　ネットワーク

オフチップ相互接続ネットワークは、今後もHPC システムの主要なボトルネックとなる可能性が高い。データセンタやクラウドにおけるAIワークロードの急増により、より高速で緊密に結合したネットワークを必要とするのはHPCだけではなくなりつつある。一方で、従来型の性能向上では、今後数年間で帯域幅は良くて4～8倍程度、遅延は現状維持または悪化する可能性がある。

HPCシステムの大規模な相互接続ネットワークでは、性能やコスト、消費電力を決定する重要な要素がシリアライザ／デシリアライザ (SerDes) である。現在、InfiniBand NDR、Omni-Path (25G)、100G Ethernetなど、1リンクあたり100Gbpsまでの高速シリアル伝送が主流で、ケーブル1本あたり4リンクが一般的である。したがって、25GbaudのNRZ変調 (または伝送損失が若干悪化する50GbaudのPAM4変調) で、最大400Gまで利用可能である。5440ビットのブロックサイズを持つFEC(Forward Error Correction) は、PAM4のビットエラー率を低下させ、約100nsのオーバーヘッドで最大15のシングルビットエラーまたは最大150ビットのバーストエラーに対応できる。今後、25/50Gイーサネットは、より低遅延の2720ビットFECに移行する可能性がある。今後10年で、InfiniBandはXDR (4xリンクで800Gbps)、GDR (1.6Tbps) を目標とし、イーサネットも同様である。

電気信号の帯域幅は距離が長くなるにつれて狭くなるが、光信号の伝送 (ファイバー使用) は距離にほぼ影響されない (0.05dB/cm)。CPO (Co-packaged Optics) とSiPho (シリコンフォトニクス)は、光と電気 (OE) の変換をLSIに近づける技術であり、SiPhoによる現在のスイッチASICの容量は25.6Tb/sに達している。CPOは、さらなる容量増加、30%以上の消費電力削減、OE変換の高速化、パッケージ密度向上等をもたらす可能性があるが、パッケージング等に研究課題が残る。

上記のSerDes、トランシーバ、ワイヤ等の基盤に基づき構成されるHPCネットワーク技術の具体例を以下に述べる。富士通のTOFUは、カーネルバイパス、RMDA、MPIをサポートし、バリアとリダクションを実行する専用のネットワーク内処理ハードウェアを必要とする。現在、ノードあたりの帯域幅は40.8GB/sまで向上し、1ホップの遅延は700ns以下まで低下している。TOFUの強みは、電力効率と低遅延だが、トポロジの制限やCPUダイの専用設計が必要な点が課題である。オープンな仕様に基づくInfiniBand (IB) は、何世代にもわたってHPCハードウェアとソフトウェアを提供している。現在、NDR (50GB/s、900ns以下のレイテンシ) は、カーネルバイパス、RMDA、MPIに加え、広範なネットワーク内処理と独自のアダプティブルーティングをサポートしている。IBの間接ネットワークは、専用のスイッチ (最大64ポート) を必要とするが、多くのトポロジ (FatTree、Dragonflyなど) を公式にサポートし、理論的には全てのトポロジをサポートできる。IBの帯域は3年ごとに約2倍のペースで向上しているが、コストの上昇、レイテンシの停滞、NIC/スイッチの消費電力が懸念材料である。Ethernetはカーネルバイパスと RMDAを備えていないが、価格、相互運用性、ソフトウェアサポートの点から、ローエンドのHPCシス

テムやデータセンタへの採用が多い。InfiniBandと比較すると、Ethernetは帯域幅では同等だが遅延はやや劣る。RDMA over Converged Ethernet (RoCE(v2)) は遅延の問題を緩和するが、採用例は今のところ多くない。EthernetのNICとスイッチによるエネルギー消費は、性能、機能、プログラマビリティに大きく依存する。HPEのSlingshotは、IBとEthernetの強みを組み合わせようとするものである。ソフトウェアはTCP/IPとRDMAを簡単に切り替えることができ、スイッチはそれに応じてパケットを処理する。また、QoS、ロードバランシング、アダプティブルーティング、輻輳管理などの機能が利用できる。現在、Slingshotは今のところ最大25GB/sのdragonflyしかサポートしておらず、加えて遅延の大きさも懸念される。Compute Express Link （CXL) とNVLinkは、同一ノード内のデバイス間の高速インターコネクトを目的として始まったが、今ではスケールアウト機能を持つ小規模なノード間接続をもターゲットにしている。Rockportはパッシブ光スイッチを提案しているが、ファイバーシャッフルのため大規模化に課題がある。EXTOLLはTOFUと似ているが、実際のシステムに使われた例は少ない。Bull eXascale Interconnect （BXI) はIBに似ているが、これも大規模な設備ではほぼ採用例がない。

ここまでネットワークシステム全体をカバーする既存の技術について述べたが、最適なシステムの構築にはSiPhoの進展やトポロジにも気を配る必要がある。現在主流となっているプラグイン光伝送の電力効率は約25pJ/bitだが、次世代HPCではSiPhoの利用により大幅な削減が期待される。すでにSiPhoを用いた光スイッチのプロトタイプでは、1pJ/bit程度を実現しており、今後はOBO (基板上モジュール) 、CPO (パッケージ内モジュール)、OE (チップ内変換) の順に技術が進展すると考えられる。必要な電力は、OBOで20pJ/bit程度、CPOで5pJ/bit程度と予想されている。一方、製造コストと光スイッチングという2つの課題がSiPhoと全光インターコネクトの普及を阻んでいると考えられる。光のバッファリングや光子のヘッダー処理、電気-光変換の低コスト化等の問題から、複雑なトラフィック制御、スイッチング等による遅延増加、リンクの活用不足など様々な欠点を持つ回線交換を選択する可能性もある。トポロジに関しては、メッシュ、トーラス、ハイパーキューブ、Clos、butterfly、dragonfly、slimfly、jellyfish等が、HPCやデータセンタをターゲットに研究されてきた。しかし、多くのHPCシステムで採用されるClosの派生型や、TOP500等のハイエンドに多いトーラスやdragonfly等、実際に採用されているトポロジは限られる。理論的には、スイッチに十分なポート数があれば全てのトポロジを実現できるが、システムスループットとコストの最適化が重要となる。

ネットワークに対する代表的なアプリケーションからの要求を考えると、3次元物理シミュレーションにおけるレイテンシ重視の小メッセージの近傍交換、AIワークロードにおける帯域幅重視の大容量削減、FFTにおける小/中サイズのオールツーオール通信等が挙げられるが、これらのすべてに適したトポロジはない。遅延に敏感なアプリは最小限のネットワークホップまたは低径ネットワークを必要とし、一方でall-to-all通信は完全な二分バンド幅を必要とし、帯域幅に敏感なアプリは通信パターンにもよるものがその中間である。現実的なアプリケーションはシステム全体の一部により実行されるためネットワークの縮小は可能だが、ジョブ間およびPFSの干渉について考慮する必要がある。

データセンタのネットワークは従来、信頼性、保守性、拡張性、柔軟性を重視し、HPCは低遅延と高帯域幅を重視してきた。現在では、AIやローエンドHPCのクラウド化が進み、これらのネットワークの機能が融合しつつある。そこでは、セキュリティ、仮想化、光スイッチ、標準化設計、相互運用性の重要性が増している。プログラマブルSmartNIC (FPGAやDPUを搭載) は、両者のニーズを満たすことが可能だが、エネルギー効率の向上やコストの削減が必要になる。

将来のHPCシステムにおけるネットワークは、CPUとアクセラレータの緊密な結合、低遅延、様々なネットワーク機能 (カーネルバイパス、HWオフロード、ネットワーク/キャッシュ結合、ネットワーク内処理、暗号化など) 等の要素をリーズナブルなコストで実現することが求められる。SiPhoとSmartNICの組み合わせは、考えられる有効な1つの方法である。他にも、CXL、NVLink、またはロックポートのソリューションなどを使用して、高帯域幅のアグリ

ゲータースイッチを備えたゼロ/低パワーの「ポッド」からなる階層型ソリューションを構築することや、低遅延で安価な電気パケットスイッチと帯域を確保する光回路スイッチを組み合わせたセミ・マルチレール・ネットワーク等も考えられる。いずれにせよ、大規模な投資と共同設計が必要となる。イーサネットと1.6Tbpsを待つだけでは、現在のシステムと比べて大きなメリットは得られないと予想される。

2.1.3 ワークロード分析

2.1.3.1 分析の概要

アプリケーショングループのベンチマーク構築サブグループは、2030年頃に求められる/期待される計算資源に基づいてアプリケーショングループの各サブグループより提出されたベンチマークの整備を行なっている。

今年度には、これらのベンチマークのうち以下の6つを用いて、次世代計算基盤のターゲットとなるワークロードの分析を行った。

表 2.1.1

名称	分野	サブグループ
SCALE-LT-kernel (SCALE)	気象気候科学ライブラリ SCALE の雷モジュール (カーネル)	気象・気候
ISPACK	流体解析カーネル	
CUBE	圧縮性流体解析	ものづくり
QWS	格子 QCD	基礎科学
qNET_kernel	密度行列繰り込み群法	
GENESIS	分子動力学	生命科学

分析の主な観点は以下の通りである。性能に関しては、富岳 (A64FX)におけるプロファイル結果に基づく。

- 記述言語
- カーネルループの構造
- 計算の種別、特徴
- 並列化の状況
- 最適化の状況
- SIMD 化
- キャッシュビジー率/ミス率
- BYTE/FLOP
- etc.

2.1.3.2 各ベンチマークの概要

2.1.3.2.1 SCALE

SCALEは気候・気象科学および計算機科学の専門家らにより開発された次世代の気象気候科学における基盤ライブラリである。本報告では、SCALEの物理モデルを利用するLightning -kernelについて述べる。このカーネルは、SCALEの物理モデルを用いて霰、氷、雹等の3次元空間における衝突回数を計算し、非誘導性電荷分離に基づく3次元モデルにより電荷密度を予測する。

プログラムはFortran90で記述され、3次元直交座標系の倍精度浮動小数点データセットに対してステンシル計算を複数回行う。プログラム内にはデータ検証部分を除いて分岐はなく、3次元のステンシル計算を行う3重ループが複数回存在している。現在のコードでは、データサイズは約16MBと小規模であり、単一ノードによる動作を想定しているため並列化はOpenMPのみである。現状では計算規模が小さすぎて、スレッド数を増やしても並列処理による高速化はあまり見られなかった。

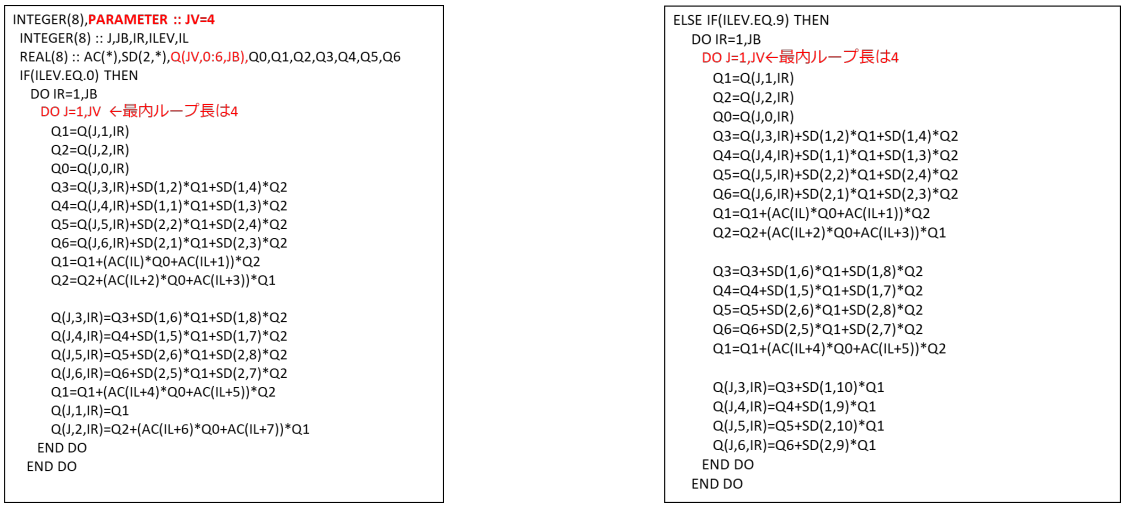
A64FXにおける1スレッド処理の結果では、浮動小数点演算性能および要求メモリ帯域はどちらもピークの２％程度となっている。キャッシュビジー率、キャッシュミス率はそれぞれL1Dが22% / 8%、L2が2% / 5%となっており、A64FXのキャッシュサイズに対する最適化は行われていないように見える。いずれにしても、現状の計算規模ではHPCベンチマークとして並列処理や通信オーバーヘッド等の評価が難しい為、問題サイズを大きくして評価する必要があると思われる。

2.1.3.2.2　ISPACK

- **概要**：簡単な流体方程式の数値計算に必要となる基本的な道具(スペクトル変換等)をサブルーチン群としてまとめたパッケージである。各サブルーチンは単独で使えることを目標として設計されている。サブルーチン群をどのように組み合わせて使うかはユーザの責任となる。流体実験等のためのサンプルプログラムは、以下の 2 種類である。
  sxpack-test: sxpack のテストプログラム：OpenMP で並列化されたパッケージ
  sypack-test: sypack のテストプログラム：MPI で並列化されたパッケージ
  言語は Fortran である。本報告では、カーネルレベルでの調査結果のみ報告する。なお、性能プロファイルには、Fujitsu PRIMEHPC FX1000 の 1 ノードを利用し、コンパイラオプションは-Kfast -Kopenmp である。
- **spack-test**：概要とコード特性：スペクトル(球面調和関数)変換を行なうサブルーチンパッケージである。球面調和関数展開の係数から格子点値、およびその逆の変換を行なうサブルーチン、また、その他の補助ルーチンから構成される。以下はテストコードの特徴であり、利用により特徴が変わる可能性がある。
  - コード最適化の度合い：主カーネルは、SIMD 化が促進する書き方になっている。SVE の命令発行率は 98%と高い。並列化効率は高いが、現在のボトルネックが並列化していないループになっている。
  - 並列化の有無：OpenMP 並列化。omp parallel で、ロードバランシングは dynamic である。
  - ループの特徴：最内ループ長が 4 と短い。全体的に、同期に費やす時間が多いと予想される。
  - カーネル部分：全体の 35%が、lxqswg 手続きである。

  [性能]: test.f90（SXPACK内での逆変換と正変換のテスト）において、1ノード48スレッド実行時の台数効果は、S2G:1.64E-02 sec（69.1GFlops):46.0倍、G2S:6.50E-03 sec (174.0GFlops):39.5倍、であり、高い台数効果が確認できる。一方、G2S処理の方が、NUMAの影響を受けやすいように見える。

  [lxqswg手続きの概要]：LXQSWG(JB,AC,SD,Q,IL,ILEV)は、図 2.1.3コードである。



```
INTEGER(8),PARAMETER :: JV=4
 INTEGER(8) :: J,JB,IR,ILEV,IL
 REAL(8) :: AC(*),SD(2,*),Q(JV,0:6,JB),Q0,Q1,Q2,Q3,Q4,Q5,Q6
 IF(ILEV.EQ.0) THEN
   DO IR=1,JB
     DO J=1,JV  ←最内ループ長は4
       Q1=Q(J,1,IR)
       Q2=Q(J,2,IR)
       Q0=Q(J,0,IR)
       Q3=Q(J,3,IR)+SD(1,2)*Q1+SD(1,4)*Q2
       Q4=Q(J,4,IR)+SD(1,1)*Q1+SD(1,3)*Q2
       Q5=Q(J,5,IR)+SD(2,2)*Q1+SD(2,4)*Q2
       Q6=Q(J,6,IR)+SD(2,1)*Q1+SD(2,3)*Q2
       Q1=Q1+(AC(IL)*Q0+AC(IL+1))*Q2
       Q2=Q2+(AC(IL+2)*Q0+AC(IL+3))*Q1

       Q(J,3,IR)=Q3+SD(1,6)*Q1+SD(1,8)*Q2
       Q(J,4,IR)=Q4+SD(1,5)*Q1+SD(1,7)*Q2
       Q(J,5,IR)=Q5+SD(2,6)*Q1+SD(2,8)*Q2
       Q(J,6,IR)=Q6+SD(2,5)*Q1+SD(2,7)*Q2
       Q1=Q1+(AC(IL+4)*Q0+AC(IL+5))*Q2
       Q(J,1,IR)=Q1
       Q(J,2,IR)=Q2+(AC(IL+6)*Q0+AC(IL+7))*Q1
     END DO
   END DO
```

```
ELSE IF(ILEV.EQ.9) THEN
   DO IR=1,JB
     DO J=1,JV←最内ループ長は4
       Q1=Q(J,1,IR)
       Q2=Q(J,2,IR)
       Q0=Q(J,0,IR)
       Q3=Q(J,3,IR)+SD(1,2)*Q1+SD(1,4)*Q2
       Q4=Q(J,4,IR)+SD(1,1)*Q1+SD(1,3)*Q2
       Q5=Q(J,5,IR)+SD(2,2)*Q1+SD(2,4)*Q2
       Q6=Q(J,6,IR)+SD(2,1)*Q1+SD(2,3)*Q2
       Q1=Q1+(AC(IL)*Q0+AC(IL+1))*Q2
       Q2=Q2+(AC(IL+2)*Q0+AC(IL+3))*Q1

       Q3=Q3+SD(1,6)*Q1+SD(1,8)*Q2
       Q4=Q4+SD(1,5)*Q1+SD(1,7)*Q2
       Q5=Q5+SD(2,6)*Q1+SD(2,8)*Q2
       Q6=Q6+SD(2,5)*Q1+SD(2,7)*Q2
       Q1=Q1+(AC(IL+4)*Q0+AC(IL+5))*Q2

       Q(J,3,IR)=Q3+SD(1,10)*Q1
       Q(J,4,IR)=Q4+SD(1,9)*Q1
       Q(J,5,IR)=Q5+SD(2,10)*Q1
       Q(J,6,IR)=Q6+SD(2,9)*Q1
     END DO
   END DO
```

図 2.1.3

2.1.3.2.3　CUBE

超大規模熱流体解析ソフトウェア「CUBE」は、COVID-19に関する高精度で大規模な「飛沫シミュレーション」において使われたことで有名だが、元々は、自動車やモノづくりの現場の様々な流体現象のシミュレーションに向けて開発されたアプリケーションであった[1]。Complex Unified Building cubE methodに由来してCUBEと名付けられている。BCM (Building Cube Method)という中橋和博教授（当時、東北大）が開発した手法をベースとして開発され、直行する構造格子を階層的なキューブとセルという構造を用いて再帰的に表現していくことにより各階層の格子の個数を固定数とし負荷分散のための均一化を図りながらも必要な精度を確保するための高精細な格子間隔にも対応できるという手法である。

CUBEの計算の対象としているのは、圧縮性・非圧縮性の流体計算であるが、より計算量が多くスーパーコンピュータの応用としても重要である圧縮性流体の計算に特に着目し、圧縮性流体の計算において支配的となる対流項（convection term）を計算するカーネルと粘性項（viscous term）を計算するカーネルの2つを今回のベンチマークに選択している。

これらの2つのカーネルは、典型的なステンシル計算のループから構成されており、計算する主な変数は5つである。次の時間ステップに行くとき大きさの異なるキューブを繋ぐための処理を行う必要があるが、今回のカーネルは時間発展のないキューブの計算のみの部分とし、キューブをつなぐための処理は含まない構成とした。各カーネルはそれぞれ単一の4重ループで構成され、最外ループはキューブ走査を行う変数Iのループ、残る3重ループはセルのIDの3次元の走査ループとなる。キューブ内のセル数は固定し、負荷分散を効率化される。今回のセル数は65としている。最外のキューブ走査の変数IのループでOpenMP並列化がされる。なお、今回のベンチマークは、MPI化はされていないが、MPIによる並列化はこのカーネルの外側で行われ、Weak Scalingとなる。データ型は倍精度浮動小数点であり、配列へのアクセスが基本となりメモリの間接参照はない。また、除算と平方根、絶対値、最大値がFortranの組み込み関数で計算されるが、それ以外にライブラリの呼び出しはない。

2つの対象カーネルの特徴を分析するために、ソースコードのレビュー、開発者へのヒアリング、プロファイリングツールによる解析を行った。ソースコードのレビューにおいては、対流項の計算カーネル部分は1164行で12点のステンシル、粘性項の計算カーネル部分は566行で16点のステンシルであり、共にループタイリングはされていないコードであった。開発者へのヒアリングを行った結果、チューニングについては富岳のプロトタイプでの性能評価のために挿入したコンパイラ向けのディレクティブがあり、変数が多いためにレジスタに効果的に収ま

るようにループ分割を行うものであるとのことであった。

プロファイリングに関しては、Exanaというツールにおいて Loop-call context treeを生成するためにx86環境で再ビルドを行い、プロファイリングを行ったことに加え、富岳の環境において富士通プロファイラを利用した性能統計情報の解析を行った。Exanaツールにおいては、変数の初期化やカーネル呼び出しの処理を含むベンチマーク全体の関数呼び出し構造とループ構造を確認するとともに、カーネル部のFLOP値やデータフットプリントを確認することができた。富士通プロファイラにおいては、ハードウェアパフォーマンスカウンタの値を読み出せるほか、FLOP/Byteの値とIPC（Instructions Per Cycle）の値を算出できることを確認した。FLOP/Byteはプリケーションのメモリバンド幅要求の度合いを示す指標である他、ルーフラインモデルで使われる演算強度にも対応する。ハードウェアパフォーマンスカウンタから得られる統計情報として特に興味深いものは、どちらのカーネルもSVE operation rate=100%であるためにベクトル化に成功していることや、対流項のL1 キャッシュミスが3.2%程度出る一方で粘性項はL1キャッシュミスが15.5%と高いことが分かった。対流項はFLOP/Byte=6.10, IPC=0.62、粘性項はFLOP/Byte=0.895, IPC=0.68であることが分かった。

[1] Ando K, Bale R, Li C, Matsuoka S, Onishi K, Tsubokura M. Digital transformation of droplet/aerosol infection risk assessment realized on “Fugaku” for the fight against COVID-19. The International Journal of High Performance Computing Applications. 2022;36(5-6):568-586.

[2] Yukinori Sato, Shimpei Sato, and Toshio Endo. 2015. Exana: an execution-driven application analysis tool for assisting productive performance tuning. In Proceedings of the 2nd International Workshop on Software Engineering for Parallel Systems (SEPS 2015). Association for Computing Machinery, New York, NY, USA, 1–10. https://doi.org/10.1145/2837476.2837477

2.1.3.2.4 QWS

QWSは富岳のターゲットアプリとして選択された格子QCDアプリケーションのカーネルライブラリ（QCD for Wide SIMD）である。詳細については、富岳コデザイン・レポート（https://www.r-ccs.riken.jp/fugaku/history/codesign-report/）を参照。性能としては、Bicgstabによるソルバが支配的。MPI+OpenMPのハイブリッド並列化がされており、A64FXでは、4ppn (1rank = 1CMG) が想定され1プロセス12スレッド実行。512bit wide SIMDを意識したデータレイアウトやカーネル部のintrinsicによる最適化がされている。メモリ再利用やL1ソフトウェアプリフェッチによる最適化などが行われている。また、通信もuTofu利用などを用いて富岳向けに最適化されており、通信と演算のオーバーラップも行われている。

ターゲットアプリとしてはL2キャッシュに載るデータサイズとして大規模ノード実行でのスケーラビリティを評価しているが、実アプリとしてはL2より大きなサイズを想定してプロセス当たりL2の8倍のデータサイズとして、4rankで評価を行った。実行パラメータはmpiexec -np 4 ./main 32 12 16 24 1 1 2 2 -1 -1 6 50。図 2.1.4が富岳における詳細プロファイラによる結果である。SIMD化率は75%と高い。演算効率としては24%程と示されているが、単精度が主なので、演算器効率としては12%。ビジー率はL1: 37%, L2: 53%, Mem:64%といずれも高いが、ミス率はL1: 10%, L2: 7%と低く、L1ミスの71%がソフトウェアプリフェッチによるものでプリフェッチの効果といえる。L2ミスの41%がデマンドミスであり改良の余地があるかもしれないが、ミス率は低くL1ビジー率への影響もあるため効果は限定的と思われる。

プログラムが複雑でプログラム上のBYTE/FLOPを直接求めることは難しいが、実行命令数からの見積もり

では、2.0。メモリ性能の実測値によるBF値は0.8であり、メモリはボトルネックにはなっていない。L2のビジー率から推定した値も2.0に近いのでL2がボトルネックになっていると推定される。

AVX512による評価についてもQWSベンチマークセット内のドキュメントに記述がある。それによると現在FSでインテルによる最適化が行われているが、現時点ではA64FXの方が2倍高速である。

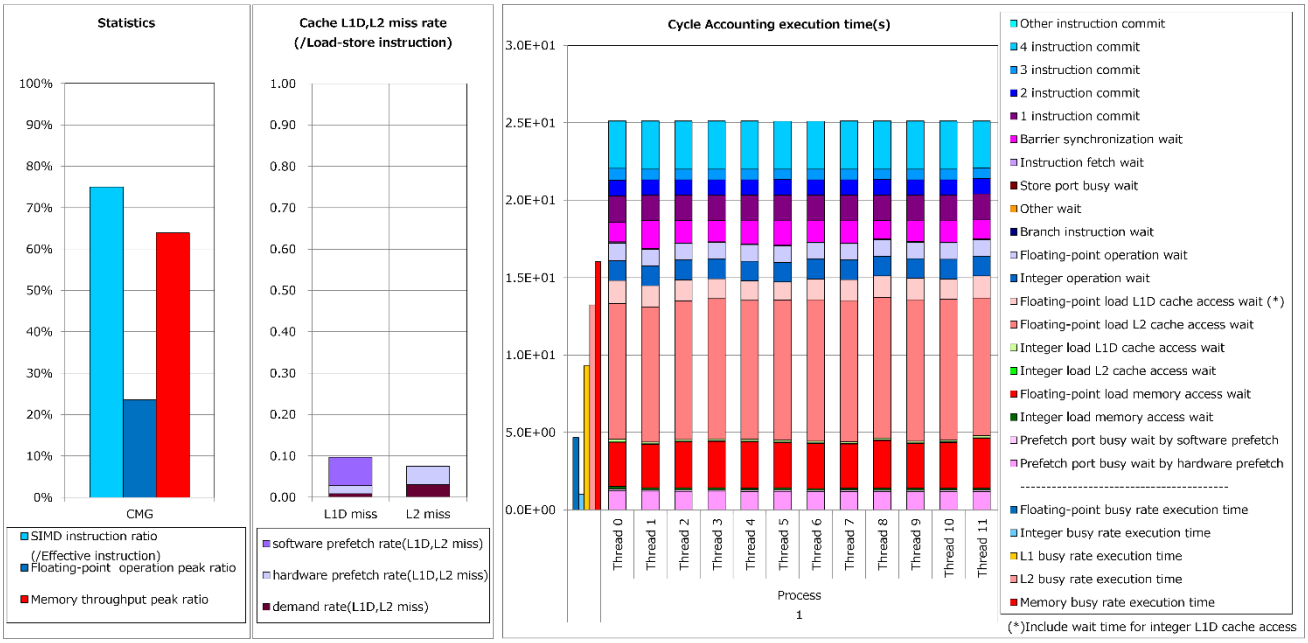


図 2.1.4

2.1.3.2.5　qNET_kernel

- **概要**：qNET_kernel とは、密度行列繰り込み群 (DMRG) 法の計算において、最も計算量の大きい部分であるハミルトニアン (疎行列) とベクトルの積を抜き出し、ベンチマーク化したものである。DMRG 法のアルゴリズムの特性から、疎行列とベクトルの積は、複数回の密行列の行列・行列積へと変換されている。ちなみに qNET とは、量子多体系の基底状態からダイナミクスまで本分野の理論研究で必要とされる様々な計算を可能にする DMRG 法によるプログラムである。多次元系の DMRG 法による高精度計算は巨大な計算コストを必要とするが、qNET を用いた大規模計算によりこれまでの手法では困難であった量子多体系の理論研究が可能となった。
- **プログラムの特徴**：qNET_kernel のソースコードの総行数は 1600 であり、Fortran90 で実装されている。先述したように、このベンチマークの計算カーネルは複数回の密行列の行列・行列積であり、図 2.1.5 のコードがその部分を示している。三重ループの最内に GEMM を呼び出すルーチンがあり、DGEMM もしくは ZGEMM を実行している。したがって B/F は DGEMM である場合、16/n (n × n 行列) となる。GEMM の実行は BLAS によるものであり、富岳 A64FX では SSL2 スレッド並列 (-Kfast,openmp -SSL2BLAMP)、スーパーコンピュータ LUMI では、OpenMP スレッド並列 (CrayBLAS) で実行されている。GPU 実行は本ベンチマークではサポートされていな

```
do ib=isbk0,iebk0
   jb=bp(1)%jbk(ib)
   if(jb.eq.0) cycle
   do i0=1,nfrx(1)
      i1=bp(1)%io(i0)
      if(i1.eq.0) cycle
      if(s0%bk(jb,1,ih)%isz(i1)*min(s0%bk(ib,1,ih)%vsize(i0),1).le.0) cycle
      do k0=1,nfrx(2)
         k1=bp(2)%io(k0)
         if(k1.eq.0) cycle
         if(s0%bk(ib,2,ih)%isz(k0).le.0) cycle

         call vclear(vout(jb),nv(i1,k1,jb),i1,k1,&
            s0%bk(jb,1,ih)%ivs(i1),s0%bk(jb,1,ih)%ive(i1),&
            s0%bk(jb,2,ih)%sizex(k1))

         zz=bp(1)%fs(i0)*bp(1)%vo(i0)*bp(2)%vo(k0)*cf

         if(allocated(pp(1)%bt(s0%bk(ib,1,ih)%ap(ad(i0,k0))%nse(1))%op)) then
            call XGEMM(chmm(1),chmm(1),&
               s0%bk(jb,1,ih)%isz(i1),&
               s0%bk(ib,2,ih)%isz(k0),&
               s0%bk(ib,1,ih)%vsize(i0),&
               zz,&
               pp(1)%bt(s0%bk(ib,1,ih)%ap(ad(i0,k0))%nse(1))%op,&
               s0%bk(jb,1,ih)%isz(i1),&
               vin(ib)%ap(ad(i0,k0))%b(:,s0%bk(ib,2,ih)%ivs(k0)),&
               s0%bk(ib,1,ih)%vsize(i0),&
               one,&
               vout(jb)%ap(ad(i1,k1))%b(:,s0%bk(ib,2,ih)%ivs(k0)),&
               s0%bk(jb,1,ih)%isz(i1))
         end if
      enddo
   enddo
enddo
```

図 2.1.5

いが、オリジナルコードでは OpenACC から cuBLAS を呼び出して実行している。MPI はベンチマーク内でサポートされていない。

- **性能評価**：ベンチマークのプログラム構造を定量的に把握するため、まずは A64FX(富岳)の 1 コア (1 スレッド) で qNET_kernel を実行し、その性能を評価した。Fujitsu Advanced Performance Profiler でプロファイルした結果を図 2.1.6 に示す。カーネルの実行時間は 13.9 秒であり、得られた性能は 46.2 GFLOPS であった。A64FX の 1 コアの理論ピーク性能は 57.6 GFLOPS であるため、80.2%の性能を達成している。ただし、qNET_kernel は GEMM による演算が殆どであるため、理論ピーク性能の 80.2%というのは若干数値が低いことは否めない。プロファイル結果から、浮動小数点ユニット(パイプライン)は 98.58%の実行効率を達成しているため、おそらく L1 データキャッシュミスによって性能が劣化していると推定される。

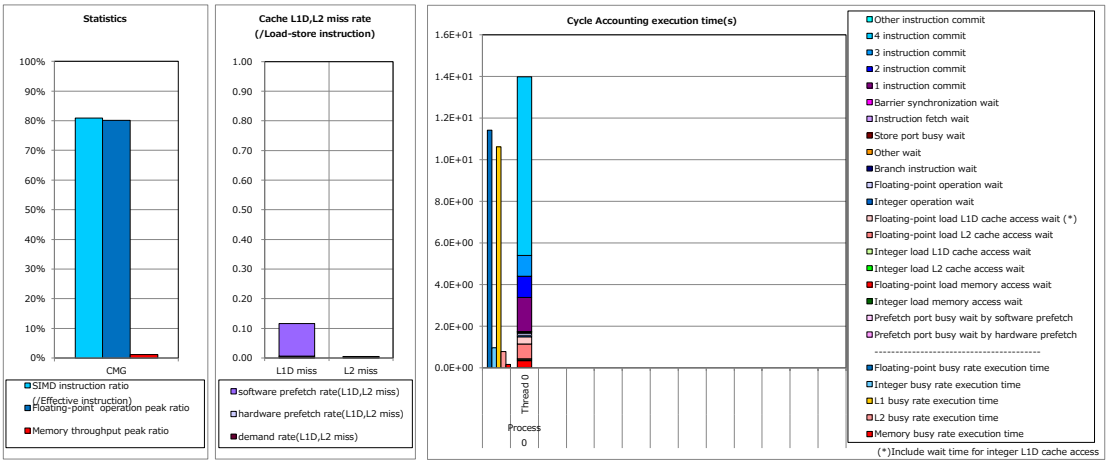


図 2.1.6

2.1.3.2.6　GENESIS

- **概要：**GENESIS は生体分子の分子動力学計算ソフトウェアであり、ポスト「京」の重点課題アプリケーションの 1 つに選定されていたアプリケーションである。Intel CPU や A64FX 等の各アーキテクチャ向けに最適化されたコードが存在するが、本 FS のベンチマークとして提供されたコード（以下、FS 向け GENESIS）は Intel CPU 向けのコードの主要な計算部分を抽出したものとなっている。コードは Fortran 90 で記述されており、主要な計算は単精度浮動小数点で行われる。

  また、オリジナルのコードではOpenMPとMPIによるハイブリッド並列化が行われているが、FS向けGENESISは1プロセス実行を想定しており、OpenMPによる並列化のみが行われている。これは、GENESISの場合、原子数の増大によって増加するのは基本的にはプロセス数であり、各プロセスの計算の特徴はほとんど変わらない（弱スケーリングする）ためである。なお、FS向けGENESISの計算規模は約10万原子を16MPIで実行した時の1プロセス相当であり、10万原子は2030年に想定される計算規模の1/10である。

  GENESISにおいてはPairlistとKernelの2つの計算カーネルが性能上のボトルネックとなる。ただし、Pairlistの実行は数ステップ（富岳開発時の想定は10ステップ）に1回なのに対し、kernelの実行は毎ステップ必要なことから、PairlistよりもKernelの方がより性能上のボトルネックとなり得る。
- **Pairlist**：短距離・非結合性相互作用を計算する原子ペアの数え上げを行う。原子の座標を表す 3 次元配列（SoA）へのアクセスと原子間の距離計算（浮動小数点加算、乗算）が主な計算内容である。A64FX における B/F は 1.4-22.5、配列の間接参照があるため SIMD 化率は 7.3%（ピーク

性能比 3.9%）と低めである。キャッシュブロッキング等の最適化は行われていないが、L1D キャッシュのヒット率は非常に高い（A64FX では 99%超）。ループの深さは最大 4（最内ループに条件分岐あり）であり、最外ループが OpenMP で並列化されている。A64FX では命令フェッチ待ち、L1D キャッシュのアクセス待ち、浮動小数点演算待ちが多い。

- **Kernel**： Pairlist で数え上げた結果を元に短距離・非結合性相互作用力を計算する。5 種類の 3 次元配列と 1 種類の 4 次元配列（SoA）を計算に使用し、浮動小数点加算と乗算が主に行われる。A64FX における B/F は 0.36-5.8、配列の間接参照がないため SIMD 化率は 73.9%と高めであるが、ピーク性能比は 5.2%と低い。Pairlist と同様、キャッシュブロッキング等の最適化は行われておらず、A64FX では L1D キャッシュのミス率が 8.9%とやや高めである（ただし、ほとんどのアクセスは L2 までにヒットする）。ループの深さは最大 2（最内ループに条件分岐無し）であり、最外ループが OpenMP で並列化されている。A64FX では L1D キャッシュと L2 キャッシュのアクセス待ちが多い。

2.1.4 アーキテクチャ検討の指針

2.1.4.1 暫定版の性能・機能要件とノードアーキテクチャ案

図 2.1.7に示す通り、暫定版の性能・機能要件である設計指針と、それを実現する見込みの高いアーキテクチャの方向性について検討を行った。設計指針としては、主にCPUによる実現される汎用性について、富岳の数倍以上の実効性能を目指す。加えて、特定のアプリケーションドメインに対しては、例えばアクセラレータなどを利用して、富岳の数十倍の実効性能を目指す。これらの実効性能を富岳と同程度のシステム電力により実現できるような電力効率を目指す。さらに、将来標準的となるであろうエコシステムとの互換性、機能の拡張性、および社会科学やデジタルツイン・Society 5.0といった新しい応用分野への対応を有するようなシステムを検討する。

これらの指針に対し有望なアーキテクチャの方向として、計算性能（FLOP）指向ではなくデータ移動（Bytes）を指向し実効効率を高める”FLOPS to Bytes”コンセプトをアーキテクチャの基本指針とする。ここでは、単なる演算器の集合体としてのピーク性能指向ではなく、演算器に対するデータ供給やノード間のデータ移動および同期と制御に着目し、システム全体としてデータ移動等のボトルネックが軽減されると同時に電力効率が向上するような方式を第一に検討する。そのために、例えば、三次元積層メモリ技術を駆使した相対メモリバンド幅の大幅な向上、シリコンフォトニクスによるリモートメモリアクセスへの高バンド幅確保、強スケーリング実行での実行効率の確保やデータフロー型の導入による低遅延実行、といった技術的特徴を備えたノードアーキテクチャを検討する。

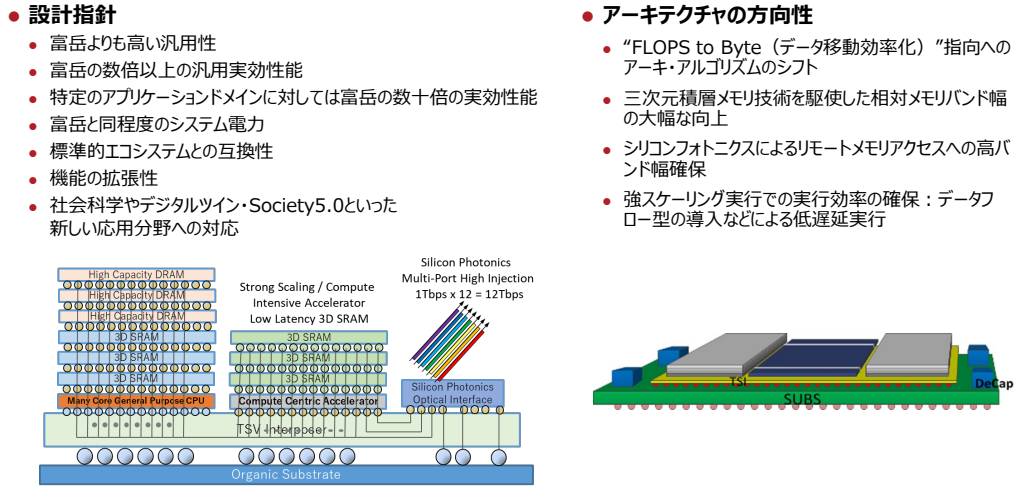


図 2.1.7 次世代計算基盤の設計指針とアーキテクチャの方向性

2.1.4.2　評価項目案と各サブグループの検討内容

サブグループとの調査研究からは複数のアーキテクチャ候補が得られる可能性があり、最終的にそれらについては評価および比較を行う必要がある。そのために、評価項目を検討した。現時点での案を以下にまとめる。

- 技術的評価指標
  - 性能に関する指標
    - 演算性能（ピーク性能、ベンチマークの実効性能）扱える計算問題の規模（ノード、システム）
    - スケーラビリティ（弱および強スケーリング）
    - 消費電力と電力性能比
    - I/O 性能
    - 設計、製造のコスト
  - 機能・特性に関する指標
    - 汎用性（ベンチマークに対する性能分布）
    - プログラミングの生産性（既存コードとの互換性、コード改変の必要量、最適化の困難度、など）
    - 既存エコシステムに対する互換性
    - 耐故障性や信頼性、可用性、保守性
    - 仮想化、クラウド利用、セキュリティ、リアルタイム処理性能等の付加的機能や拡張性
- 戦略的評価指標
  - 競合技術との差別化、新技術開拓の度合い
  - 開発計画の確度（計画通り進まないリスクなど）
  - 技術の継続性
  - 商用展開、技術展開の可能性（販売が期待できる市場、技術展開先の候補など）
  - 日本が独自に開発・保有すべき技術とその強み、日本の技術との連携や日本の産業への波及効果
  - 日本の産業競争力強化や経済安全保障への貢献
  - 国際連携・戦略的互恵関係の可能性や発展性
  - アカデミアや産業界における国内での人材確保や人材育成への波及効果
  - ハードとソフトを開発できる国内技術と当該コミュニティの維持

評価指標案は大きく「技術的評価指標」と「戦略的評価指標」の大項目に分かれており、前者はさらに「性能に関する指標」と「機能・特性に関する指標」から成る。「性能に関する指標」は演算性能、スケーラビリティ、電力性能比など、数値化可能な指標であり、システムの性能を表す。「機能・特性に関する指標」は汎用性、生産性、既存エコシステムに対する互換性などの、機能や特徴を表し数値では比較し辛い項目から成る。それぞれについては改めて定義を必要とする項目もあるが、例えば汎用性については、与えたベンチマークプログラムに対する実行可能性や実効性能の分布により評価可能と考えている。

「戦略的評価指標」は、性能や機能といった技術的な側面ではなく、開発の戦略や日本および社会に対する必要性という観点での評価軸である。例えば、競合技術との差別化の度合いや、開発計画の確度・リスク、開発技術のその後の継続性を含む。また、産業競争力の強化の可能性や、経済安全保障への貢献、および国際連携・戦略的互恵関係の可能性や発展性、人材育成への波及効果なども含まれる。これらの評価項目案について、最終的には、評価の優先順位やKey Performance Indicator (KPI) を整理した上で、候補アーキテクチャについて比較評価を行うことを想定している。

2.2 アーキテクチャ調査研究サブグループ 1（理研）

2.2.1 調査研究の概要

強スケーリング性能を有する、準汎用～専用加速装置アーキテクチャの検討を行っている。理化学研究所 生命機能科学研究センター 計算分子設計研究チームにおいて蓄積がある分子動力学シミュレーション専用計算機を題材に、まず専用加速装置としての開発を実施し、それをベースとしてその汎用化・準汎用化された演算加速装置の可能性を検討する。特に、分子動力学シミュレーション専用計算機の演算能力の中核であった非結合力計算の加速に加え、同期機構・メモリ内演算などの周辺回路における加速を重点的に検討すると同時に、これまで限定的にしか行っていなかった結合力計算の加速の検討を行っている。

今年度は、性能評価に向けた具体的な回路設計を中心に行った。設計にあたっては、Chisel言語を使用し、極力パラメータ化して設計を行った。分子動力学シミュレーション専用計算システムで必要になる要素は以下である。

- 非結合力計算加速パイプライン
- 結合力計算・粗視化分子動力学計算向け汎用コア
- メモリ
- イベント制御部
- オンチップネットワーク
- 長距離力計算加速部
- オフチップネットワーク

これらの内、長距離力計算加速部・オフチップネットワークを除いた部分の開発を本年度は進めた。これらの部分は重要ではあるが、今回特に重要な更新で、汎用化の可能性が高いイベント制御回路および結合力計算・粗視化分子動力学計算向け汎用コアの評価には不要なためである。ほぼ必要な要素の設計を終え、ソフトウェア作成・小規模な系でのテストを開始する段階に到達した。来年度は、引き続き設計を進め長距離力計算加速部・オフチップネットワークの開発を進める。また、現時点でもカットオフ計算に基づくMDシミュレーションが可能な状態になっているので、まず1ユニットでのテストから複数ユニットをNetwork-on-chipで接続したシステムの評価までを並列で進め、シミュレーションソフトウェアの作成、ハードウェア・ソフトウェアのデバッグ、性能評価を行う。その後長距離力計算加速部・オフチップネットワークを統合し、複数LSI（FPGA）上でのシステムで長距離力計算を含めた評価を行う。これに基づき、汎用計算機に付加する専用アクセラレータの可能性、準汎用化の可能性の検討を進める。

2.2.2 アーキテクチャの検討状況

2.2.2.1 アーキテクチャ検討の方針

強スケーリング性能を有する、準汎用～専用加速装置アーキテクチャの検討を行う。理化学研究所 生命機能科学研究センター 計算分子設計研究チームでの蓄積がある分子動力学シミュレーション専用計算機を題材に、まず専用加速装置としての開発を実施し、それをベースとしてその汎用化・準汎用化の可能性を検討する。

2.2.2.2 検討中のアーキテクチャ・テクノロジ・システムスタックなど

汎用計算部に付加する演算加速装置を検討している。今回の評価では、演算部のみでなく、同期機構・メモリ内演算などの周辺回路における加速を重点的に検討する。また、専用計算機の演算能力の中核であった非結合力計算の加速に加え、これまで限定的にしか行っていなかった結合力計算の加速の検討を行った。

2.2.3 調査結果

2.2.3.1 全体設計方針

今年度は、性能評価に向けた具体的な回路設計を中心に行った。性能評価、汎用化可能性については2023年度に行う。設計にあたっては、Chisel言語を使用し、極力パラメータ化して設計を行った。これは、最適化のために加え、汎用化にあたっての設計流用を行いやすくするためである。

最終的なシステムで必要になる要素は以下である。

- 非結合力計算加速パイプライン
- 結合力計算・粗視化分子動力学計算向け汎用コア
- メモリ
- イベント制御部
- オンチップネットワーク
- 長距離力計算加速部
- オフチップネットワーク

これらの内、長距離力計算加速部・オフチップネットワークを除いた部分の開発を本年度は進めた。これらの部分は重要ではあるが、今回特に重要な更新で、汎用化の可能性が高いイベント制御回路および結合力計算・粗視化分子動力学計算向け汎用コアの評価には不要なためである。

2.2.3.2 イベント制御部の調査結果

2.2.3.2.1 MD のステップシーケンサの概要

まず、中粒度でのタスクを駆動するステップシークエンサーに必要な機能を検討した。MDの 1 ステップは大まかに以下のようになる（velocity verlet法、並列化前提）

1. 速度・座標の更新
2. 座標の送受信
3. 力の計算
   (a) 二体相互作用
   (b) 結合力
4. 力の送受信
5. 速度の更新

1, 3b, 5 は汎用コア、3aは非結合力計算加速パイプライン、2,4はDMAコントローラとネットワークで処理される。ノード間のタイミングの違い等によって最適な並列実行順序は変動するので、これに対応するためイベント駆動ハードウェアを実装する。イベント駆動回路は、各フェーズ間の依存関係、フェーズにより駆動されるモジュール処理、フェーズが依存するモジュール処理の設定に基づき、処理を駆動する。図1にイベント駆動処理の流れの概念図を示す。全体の計算を中粒度（例えば一つの結合力計算、1セルペアに対する非結合力計算などを想定）の処理（図 2.2.1の”Sequence”）に分解し、各処理間をイベントで繋いでいく形で1ステップの計算を記述する。

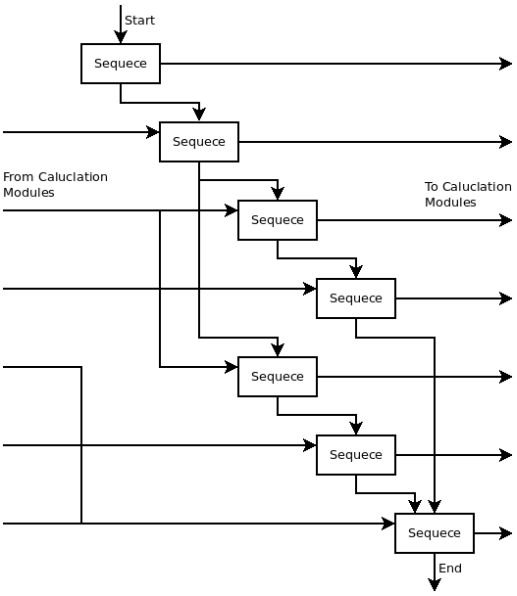


図 2.2.1 イベント駆動処理の流れの概念図。Sequence はひとまとまりの処理

2.2.3.2.2　結合力イベント駆動

結合力の計算は汎用コアで実行するが、計算対象の粒子の組み合わせが多様で、必要な粒子情報が揃うタイミングは他ノードの処理・通信に依存して変動するため、粒子情報の受信を監視し計算可能となった結合力計算から順次汎用コアへ要求するハードウェアを実装した。機能の概要は以下である。

1. 結合力計算に必要な粒子データの受信を監視する
2. 結合力毎に必要とする粒子の受信状態を更新する
3. 粒子が揃った結合力イベントを駆動する
4. 駆動された結合力イベントに対応する計算関数・引数を渡して汎用コアを駆動する

このために、粒子番号からどの結合力に必要な粒子かを特定する回路（図 2.2.2）、必要な粒子が揃った結合力イベントを駆動する回路(図 2.2.3)、駆動されたイベントから汎用コアで実行する関数と引数を生成する回路（図 2.2.4）などをなるべく汎用的な形で設計した。また、計算に必要な粒子データのローカルメモリ（キャッシュ）へのロード、計算結果のメモリへのストア・他ノード送信もイベント駆動回路内にハードウェア実装し、汎用コアは基本的には計算関数の実行のみ処理すればよいようになっている。

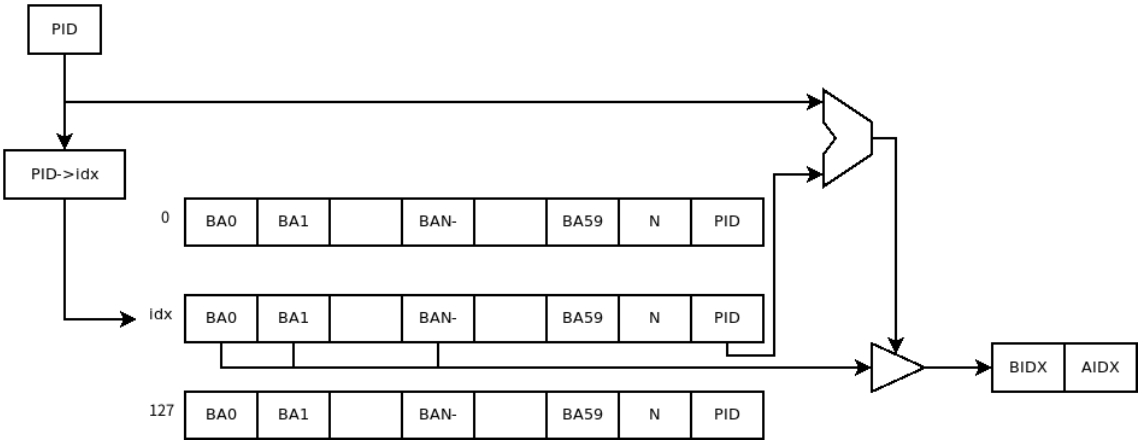


図 2.2.2 粒子番号からどの結合力に必要な粒子かを特定する回路のブロック図

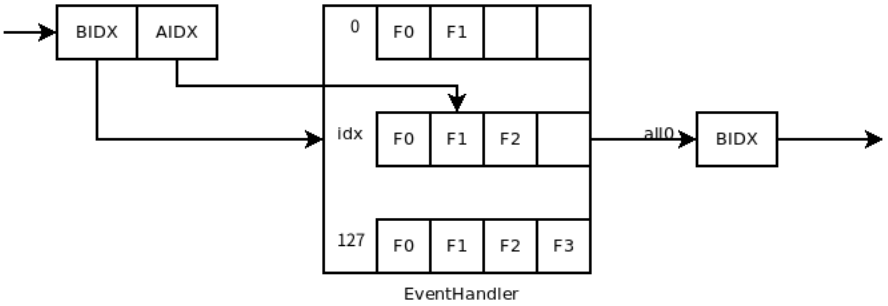


図 2.2.3 必要な粒子が揃った結合力イベントを駆動する回路ブロック図

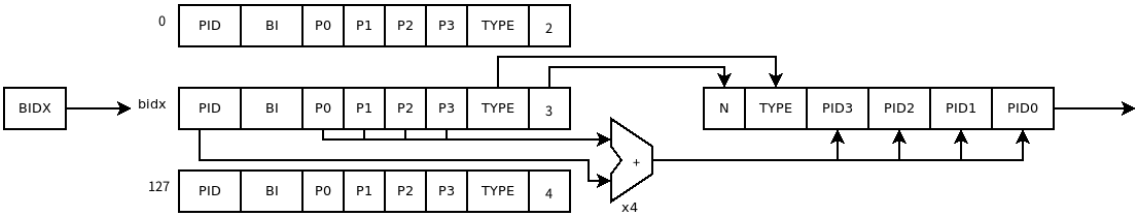


図 2.2.4 駆動されたイベントから汎用コアで実行する関数と引数を生成する回路のブロック図

2.2.3.2.3　非結合力計算加速専用パイプライン駆動

専用パイプラインは計算対象を空間分割したセルのペアで計算を行なう。セルはノード間に分散しており、結合力計算と同様に計算可能となるタイミングはノード負荷や通信で変動する。このため、セルの状態監視と通信、セルペアが揃ったときに計算を自動的に駆動するハードウェアを実装することで極力効率を上げるよう工夫した。本機能の概略は以下である。

1. セルの使用可能状態の通信
2. 計算可能なセルペアを検出、ロード・計算を駆動
3. 専用パイプラインのセルロード、ストア（計算結果）要求を実行するDMAC

2.2.3.3　結合力計算・粗視化分子動力学計算向け汎用コア部の調査結果

まず、Chisel言語で書かれた演算・論理ライブラリRial (RIKEN Arithmetic and Logic library)の拡張を行った。具体的には以下の項目を実施した。分子動力学計算だけでなく、機械学習の加速に向けた機能も一部含まれる。

- 浮動小数点積和演算器
- 逆数・平方根・逆数平方根・三角関数・逆三角関数・指数関数・対数関数を含む特殊関数の実装・改良
- 浮動小数点数を出力する乱数生成機の実装：整数値の乱数を出力する回路から、それぞれ
  - 一様 [1, 2) の範囲に変換するもの
  - 一様 [0, 1) の範囲のあらゆる normalized number に変換するもの
  - 標準 Gaussian に従う乱数に変換するもの

  を実装
  - 数学関数の一部が前処理・後処理で使用する乗算機を一つの乗算機に統一し回路規模を低減
  - 数学関数モジュールが必要な関数のみをインスタンス化できるよう変更

- Softplus 関数の実装（FP32, BF16）
- Sigmoid 関数の実装（FP32, BF16）

このライブラリを活用しつつ、結合力計算・粗視化分子動力学計算向け汎用コア(General-purpose core, 以下GP-core)の実装、およびアセンブラの作成を行った。実装の特徴を以下に示す。

- FPU の段数（デフォルト 4 段）に合わせた数のスレッドに順に命令を発行することで、Out of Order なしに FPU を全段稼働させられる(Hyperthreading)
- 分岐命令の条件チェックと分岐先アドレスの設定をスレッドの命令発行間隔に合わせることで、投機的実行なしに分岐によるストールを排除
- 命令の順序変更や投機実行がないために（GM などとの相互作用を除き）パフォーマンスが予測可能、データ依存もアセンブラの段階で判定
- GM への r/w、DMA の await ではストールが発生する
- Rial によって物理シミュレーションに必要な代表的な数学関数が全て 2 命令分のサイクル数で実行可能、具体的には：1/x, sqrt, 1/sqrt, sin, cos, acos, atan2, log, exp
- 各スレッドに対して 8 命令保持する命令キャッシュにより命令フェッチステージを（分岐命令がブロックの最後の命令だった場合を除いて）1 サイクルに
- 命令のビットパターンと引数を定義するソフトウェアライブラリを実装し、それを使って Chisel でのデコードに使用する命令リストを自動生成し、また命令をバイナリに変換するアセンブラを実装
- アセンブラには複数の命令に展開される高水準な命令をいくつか追加（32bit 即値、ラベルによる分岐）

2.2.3.4　メモリ部の調査結果

メモリ部については、MDGRAPE-4Aの設計を踏襲し、Chiselで再設計すると同時にパフォーマンスの改善を図った。メモリ側にセル管理機能、力などの積算機能を持たせることにより、複数プロセッサ間での同期コストを削減している。また粒子IDからのアクセスを可能にする粒子管理機能を有し、アクセス遅延を削減している。実装の特徴・仕様を以下に示す。

- アドレス下位ビットで担当範囲を分割する複数のメモリバンクを用意し、複数のポートからの読み書きを（対応バンクが異なる場合は）同時処理可能
- 各 word に 1bit の parity bit を付与し、誤り検出が可能
- 各 qword にアトリビュートを付与し、読み込み時にアトリビュートが異なる値をマスク可能
- 書き込みデータは qword 中の word 単位でマスク可能
- パイプライン化されたアキュムレートモジュールを実装し、値の加算代入をメモリ上でサポート
- 読み込み書き込み両方でバースト転送をサポート
- バースト転送時に一定パターンでの qword のスキップをサポート
  - パターン 0: 1111’1111
  - パターン 1: 0101’0101
  - パターン 2: 0011’0011
  - パターン 3: 0001’0001
- バースト読み込みの際に、常に全バンクへリクエストを送れるよう先読みパターンと結果の並べ替えを実装し、レイテンシを低減
- メモリをブロックに切り分け、ブロックをセルに割り当てることでセル単位でのアロケーションと、セル単位でのバーストアクセスをサポート
- セルへの粒子の追加と連動して更新される粒子マップを実装し、粒子のアドレスを ID から検索可能

2.2.3.5　非結合力計算加速パイプラインの調査結果

非結合力計算加速パイプラインについては、MDGRAPE-4Aの設計を踏襲しており、これをChisel言語で書き直した。図 2.2.5に非結合力計算加速パイプラインのブロック図を示す。但し、MDGRAPE-4Aでは固定シーケンスで駆動していたが、本設計では2.2.3.2.3のイベント駆動で説明したようにダイナミックに駆動している。そのため、パイプラインの粒子レジスタについては、キャッシュとして再設計を行った。

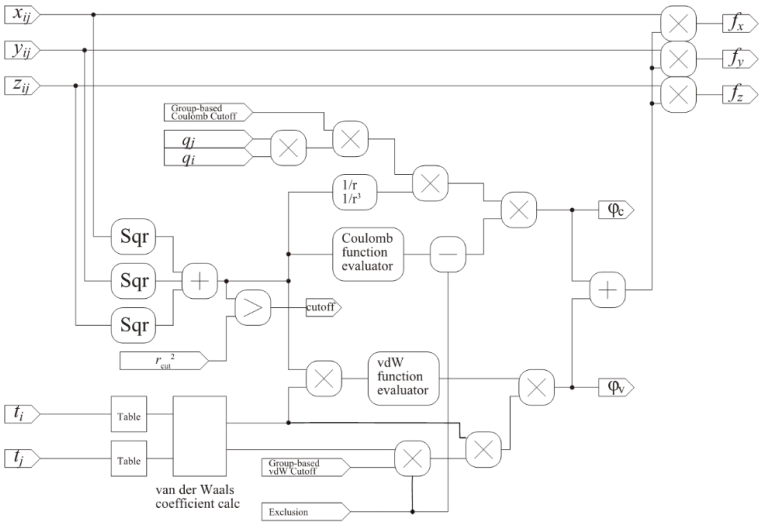


図 2.2.5 非結合力計算加速パイプラインのブロック図

また、非結合力計算加速パイプラインを汎用演算アレイに実装する可能性について、サブグループ2の富士通株式会社に協力し検討を進めている。

2.2.4　今後の課題と計画

今年度はデバッグ完了していないが、ほぼ必要な要素の設計を終えることができ、ソフトウェア作成・小規模な系でのテストを開始する段階に到達した。来年度のMD専用機としての評価計画は以下である。

- 1 ユニット（メモリ・汎用コア・専用パイプライン・イベント駆動回路）を用いたシミュレーションソフトウェアの作成、ハードウェア・ソフトウェアのデバッグ、性能評価
- 複数ユニットを（別プロジェクトで開発を進めた）Network-on-chip で接続したシステム上でのシミュレーションソフトウェアの作成、ハードウェア・ソフトウェアのデバッグ、性能評価
- FPGA 上での実装と稼働
- 複数 FPGA を用い、Off-chip network も含めたシステム上でのシミュレーションソフトウェアの作成、ハードウェア・ソフトウェアのデバッグ、性能評価
- 長距離力計算加速部の開発
- オフチップネットワークの開発
- 並行して、GP-core 向けのコンパイラの実装を検討する。

これらの性能評価をベースに、汎用部に付加する専用アクセラレータの可能性、準汎用化の可能性の検討を進める。

2.3 アーキテクチャ調査研究サブグループ 2（富士通）

2.3.1 調査研究の概要

富岳の後継機では富岳以降の技術動向の変化に対応した新しい技術の導入が必要になる。最も重大な技術動向の変化は「Mooreの法則の終焉」と呼ばれる半導体プロセス微細化ペースの鈍化である。微細化ペースの鈍化によりアーキテクチャに起因する面積・電力あたり演算性能の差の「年換算」が拡大し、面積・電力効率の良い特定処理向け演算器すなわちアクセラレータの必要性が増大する。ペース鈍化の主因は半導体メーカーの投資額増大による半導体製造コストの増加である。一方でペースが鈍化しつつも微細化は継続し回路規模は向上するので、設計コストの増加も継続する。これらの変化に対応する新しい技術としてチップレット技術が有望である。チップレットとは小さなダイのことで、大型の単一ダイの代わりに複数のチップレットでプロセッサを構成し、チップレット単位で試験することで歩留りを向上する。さらにチップレットの用途によって異なるプロセス技術を適用することによる製造コストの削減や、開発済みチップレットを他製品に再利用することによる設計コストの削減も可能になる。

富岳後継機のアーキテクチャについて、ノードおよびシステムの観点で技術動向調査とアーキテクチャ検討を行う。技術動向調査では2028年度出荷開始と2030年度稼働開始を想定し、利用可能な半導体プロセス・パッケージング技術、プロセッサ技術、アクセラレータ技術、メモリ技術、高速伝送・高密度光伝送技術、高帯域入出力技術、ストレージ技術を調査する。アーキテクチャ検討では幅広いアプリケーションを加速するノードアーキテクチャと、計算科学およびデータ駆動科学を加速するシステムアーキテクチャを検討する。ノードアーキテクチャでは三次元実装とチップレットを含む先進パッケージング技術を活用し、プロセッサ、アクセラレータ、高帯域メモリを統合するヘテロジニアスな密結合アーキテクチャを検討する。システムアーキテクチャでは高密度光伝送と高帯域入出力を活用し、高性能な超並列計算機を構成するインターコネクトおよび異種システム間の連携を高める高速データ共有システムを検討する。本年度は技術動向調査およびアーキテクチャ初期検討を実施した。

ノードアーキテクチャの技術動向調査では半導体プロセス技術の性能、電力、面積、製造コスト、設計コストについて動向を調査し、富岳で使用した7nm世代比のパラメータを抽出した。さらにメモリ技術動向を調査し、利用可能なデバイス仕様をパラメータ化した。また、アーキテクチャ検討結果との比較のためにプロセッサおよびアクセラレータの技術動向を調査した。

ノードアーキテクチャの検討では、次年度以降のアプリケーションとのコデザインでは様々なシステム設計方針に適合するノードアーキテクチャの考慮が必要なことから、プロセッサ、アクセラレータ、メモリについて、それぞれ特徴の異なる３タイプを検討し、組み合わせ可能なアーキテクチャモデルを確立した。プロセッサでは富岳と同様にモノリシックなダイで構成するType-C、チップレットの再利用性を活用して設計コストを抑えたType-B、チップレットの低製造コストを活用して大型パッケージを実現するType-Aをモデル化した。アクセラレータでは密行列計算に特化した固定機能型のType-C、密行列計算に加えて計算科学で使用頻度の高い特定処理も加速するバランス型のType-B、原理的には幅広くアプリケーション一般を加速可能な再構成可能型のType-Aをモデル化した。また、アクセラレータを搭載しないType-0も選択可能とした。メモリではコストの低い安いモバイルメモリを活用するType-C、富岳と同様に高帯域メモリを使用するType-B、SRAMの三次元積層で非常に高い帯域を実現するType-Aをモデル化した。このモデルではプロセッサ、アクセラレータ、メモリの順でタイプを列挙して、ノードアーキテクチャの組合せを表記する。例えば富岳と同様のアーキテクチャはType-C0Bとなる。また、本モデルに技術動向調査で抽出したパラメータを適用し、いくつかのシステム設計方針の制約下でノードアーキテクチャの特性を比較分析する初期評価を行った。次年度以降は本モデルとパラメータをアプリケーション性能評価に活用し、各モデルにフィードバックしつつ、アーキテクチャを探索する。

システムアーキテクチャの技術動向調査では従来の高速伝送、光伝送技術の動向調査に加え、新興の高密度光伝送技術の開発動向を調査した。また、ストレージ、ディスアグリゲーテッドメモリ技術の動向調査に加え、並列ファイルシステム等のデータ共有システムの技術動向を調査した。

システムアーキテクチャの検討では、ノード数に応じた適切なネットワークトポロジに対し、高密度光伝送技術を適応するインターコネクト構成の初期検討と、各種高速データ共有システムのハードウェア構成の初期検討を実施した。次年度以降はアプリケーション性能評価に活用可能なモデル化と、アプリケーションとのコデザインを実施する。

2.4 アーキテクチャ調査研究サブグループ 3（インテル）

2.4.1 調査研究の概要

国立研究機関を中心とした研究開発だけでなく民間業務でも広く利用されるスーパーコンピュータ・システムを実現するには高性能なハードウェア(確かなロードマップに基づくCPUコア、GPUアクセラレータ、プラットフォーム、ネットワークのさまざまな世代)を融合し、エコシステム全体で複数のコンポーネントを統合する必要がある。そして効率の良いソフトウェア環境とそのエコシステム自体をサポートするアクティブなパートナーが必要である。

システムオンチップ（SoC）、ノード、プラットフォーム、システムアーキテクチャおよびアプリケーションに関して、基本的なパラメータ(メモリ帯域幅、FLOPSあたりの帯域幅)の初期的な感度分析を開始した。また、コンパイラやソフトウェアプログラミング環境において使用する標準的なミドルウェアの効率的な実装を検討し、富岳NEXT向けの高性能なストレージスタックとしてDAOSの検証を開始した。

2.4.1.1 アーキテクチャの調査

現在、技術的難易度の高さに応じた3つの方針を検討している。それらの方針の概要は以下の通りである。

- 方針 A: 非常に高い技術的な課題を伴う、新しい非公開の構成。劇的なパフォーマンス、およびワット当たりパフォーマンスの向上を目指し、複数の技術に渡って限界を押し広げる。
- 方針 B: 汎用 CPU コアチップレットと汎用アクセラレータ・チップレットを組み合わせた統合 XPU 構成。内部の通信、および外部との接続・ネットワークを扱うスーパー・ソケット上に実装される。
- 方針 C: 従来の CPU と外部のアクセラレータを組み合わせる現在のディスクリートなアーキテクチャの延長にある機能拡張。

この中で方針A、Bの両方が最も有望な選択肢となる。方針Cに関しては主に比較対象として使用される見込みであり、ターゲットとする時間軸において説得力を持つ選択肢になる可能性はほぼ無い。

方針A～Cのすべてにおいて、基本となる仕様は理研の示す野心的な目標スペックを満たしている。

- 2～3EB/s、もしくは富岳の 15~25 倍の範囲の総メモリ帯域幅
- 低精度の場合は 100EFLOPS、もしくは富岳の 50 倍のピーク FLOPS 目標
- 約 30MW、もしくは富岳の 1.5 倍の平均総システム電力

以降の各節で解説する通り、さまざまなワークロードにおける初期感度分析を使用してこれらの数字を絞り込み、さまざまな実際の(及び将来の)アプリケーションにおいて性能を富岳の30～100倍向上させる最適化された仕様を提供することに焦点を当てていく。

また、既に実績のあるインテルのx86 CPUとインテルのアクセラレータを基本とし、日本独自の汎用プロセッサや汎用アクセラレータ・チップレットの組み合わせも検討している。これは、UCIe(ユニバーサル・チップレット・インターコネクト・エクスプレス)などの標準インターフェースを用いて、標準的なシステム・オン・パッケージ(SoP)の概念の中で統合するものである。これにより、高度な研究から民間まで幅広い市場に展開できる構成を目指す。

2.4.1.2 コンパイラ及びソフトウェアプログラミング環境の調査

oneAPI イニシアチブの一部として、インテルはオープンでクロスベンダなエコシステムの構築を支援している。これにはオープンソース技術とオープンコミュニティフォーラムによって管理される業界標準を利用する。CPUやアクセラレータ・チップレット上で動作する標準的なプログラミングモデルを提供するために、oneAPIコンパイラ(SYCLやOpenMPのようなプログラミング言語を含む)、高レベルオープンソースライブラリ(例：oneAPI Deep Neural Network LibraryやIntel Neural Compressor and Intel DPC++ Compatibility toolのような)オープンソースプロジェクトの利用を提案する。

2.4.1.3　高性能ストレージ及びラージスケールファイルシステムの調査

インテルはオープンソースストレージ基盤としてDAOSを提案する。元々はx86_64向けに開発が行われてきたが、本調査ではARM64上でのDAOS検証を行った。アーキテクチャ調査での重要な焦点はfabricの統合(ハードウェアとソフトウェアを含めた光電融合の実現)である。システムソフトウェアレベルでは、継続性のあるHPC/AIフレームワークの統合が生産性と性能において重要である。スケールアウトストレージシステムに計算ストレージ機能を追加することはクライアントへのデータ移動を最小化し、ストレージハードウェア性能の世代間の性能改善を超えた性能を実現するために必要である。我々は、これらの能力を共同で前進させるための日本の産学とのパートナーシップを歓迎する。

2.5 アーキテクチャ調査研究サブグループ 4（AMD）

2.5.1 調査研究の概要

2030年の導入を目指す、次世代計算基盤における計算ノードのアーキテクチャの調査・研究において、AMDは特別な立場にいる。それは、米国政府およびHPE社との共同設計により、世界最速のスーパーコンピュータ「Frontier」を構築済みであることから実証できる。「Frontier」には、AMD EPYC™ プロセッサ、AMD Instinct ™ アクセラレータ、ROCm™ オープン・ソフトウェア・プラットフォームが採用されている。加えて、今年発表予定のスーパーコンピュータ「El Capitan」では、AMDのハードウェアとソフトウェアが活用され、2 Exaflopsを超える高い性能が期待されている。

こうしたハイパフォーマンス・コンピューティング (HPC) に関する専門知識を活用し、AMDは富岳NEXTのプロセッサと計算ノードのアーキテクチャの候補技術を選定した。また、目標とする導入時期に向けて、利用可能な技術の予測を行った。半導体のプロセス技術の進歩が鈍化し、シリコンのコストが上昇している中、パッケージングおよび積層技術はますます重要になってきている。こうしたAMDの先端技術は、アーキテクチャのカスタマイズ化や、多様な市場の要求に沿った技術提供に貢献している。

CPU、アクセラレータ、メモリ、インターコネクトを密結合した計算ノードのアーキテクチャ候補をAMDのパッケージングと3次元積層技術を用いて検討した結果、以下の４つに分類することができた。

1. CPUのみを考慮したアーキテクチャ
2. 今日の AMD Instinct™ アクセラレータをベースとしたアーキテクチャ
3. 理化学研究所のProcessing Element (PE) に着想を得たアーキテクチャ
4. 富岳NEXTの研究開発によって促進される技術革新を用いたアーキテクチャ

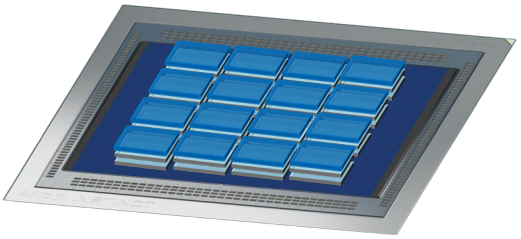


Illustration of a conceptual node architecture

図 2.5.1 計算ノードの提案アーキテクチャ

中間調査の結果、既存の富岳を大きく上回る性能と電力効率を実現するためには、アクセラレータの活用が不可欠であることが分かった。本中間報告書では、計算ノードの構成要素とその構成比率が、性能、メモリ帯域、メモリ容量、電力効率にどのような影響を与えるかを考察する。

AMDは競争力のあるハードウェアとソフトウェアを提供するために、革新的な技術を協業しながら採用してきた実績がある。アプリケーション、システム・ソフトウェア、ライブラリをも含んだ共同設計は、次世代の「研究デジタルトランスフォーメーション」の実現に寄与すると自負している。

2.6 アーキテクチャ調査研究サブグループ 5（NVIDIA）

2.6.1 調査研究の概要

<富岳NEXT 調査研究>

富岳NEXTにおいて、我々は以下の提案をコンセプトとする。

1. 多様化するワークロードや予測不能な地球環境、社会環境の変化に対応するため、NVIDIA が考える製品や技術を基に **GPU-CPU-DPU を統合した柔軟性に優れたデータセンタの構築を提案**する
2. 2030 年世代を見据え、特定分野のアプリケーションやアルゴリズムに最適なハードウェアやソフトウェアをどう実装すべきか、理研とのコデザインの中で議論する。ただし予測不能な数年先に向けたシステムの設計となるため、議論は調査研究以降、最終段階まで継続する
3. 限られた消費電力の中で必要な**アプリケーション性能を最大化することが最重要課題**であり、理研とのコデザインの中で 10 年後も引き続き重要と思われる実アプリケーションをベースに今後のワークロードも見据え、電力効率の向上について議論する
4. 日本国内パートナーや日本固有技術との連携について引き続き可能性を模索する

＜技術背景＞

計算科学とデータ科学の統合により、スーパーコンピュータは新たな時代に突入している。これらの技術進化は、富岳NEXTが導入される2030年に向かって想像を超える速度でさらに進化し、計算基盤の重要性は益々高まると考える。一方、半導体技術はムーアの法則の限界に達し、一定の消費電力での部品レベルの生産性向上率は不可能となっている。

我々NVIDIAは、20年以上にわたりGPUを開発し、プロセッサの理論性能およびアプリケーションの実行性能の継続的な向上に取り組んできた。最近では、GPU単体の性能向上のみならず、CPU-GPU-DPUをシステムとして統合するための研究開発にも注力し、多様化するアプリケーションの性能向上を実現する取り組みに注力している。

<2030年に向けてのロードマップ>

NVIDIAはこれまで2年ごとに新しいアーキテクチャを開発し、前世代の2倍以上の性能向上を実現してきた。今後は、CPUやDPUも含め、それぞれのハードウェアアーキテクチャを2年ごとのサイクルで刷新していくことを継続する予定である。

アプリケーションの実行性能については、ハードウェアの改善のみならず、ライブラリやツールなどソフトウェアの改善によって、同じアーキテクチャの中でも性能向上を実現している。このことが示す通り、NVIDIAはハードウェアによる理論性能の向上だけでなく、アプリケーションの実効性能の向上に対しても積極的にコミットする。

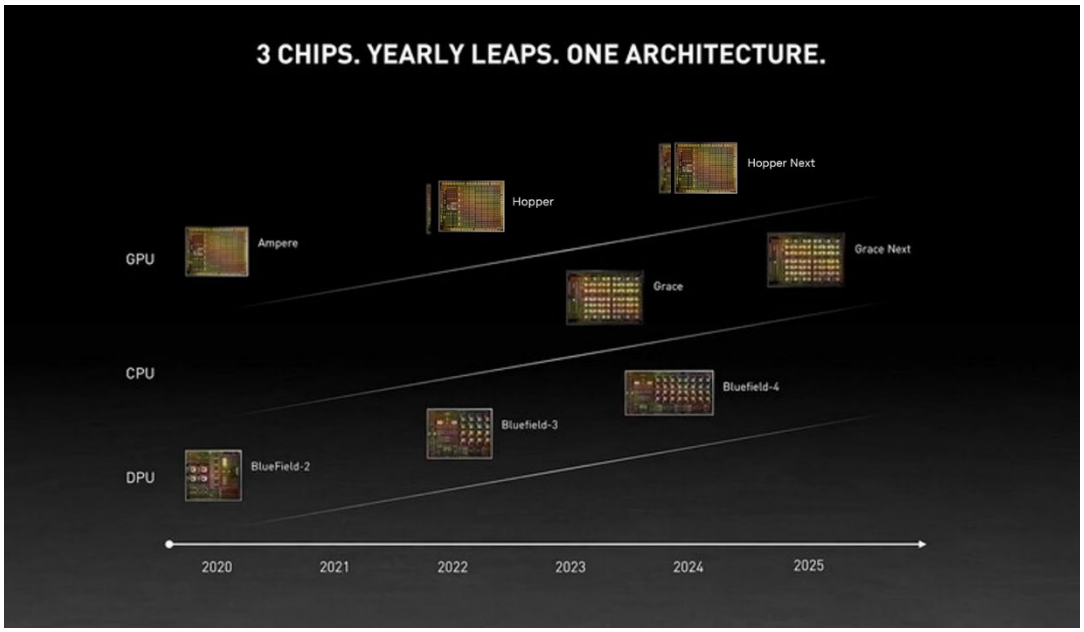


図 2.6.1 NVIDIA ハードウェアロードマップとアプリケーション実行性能

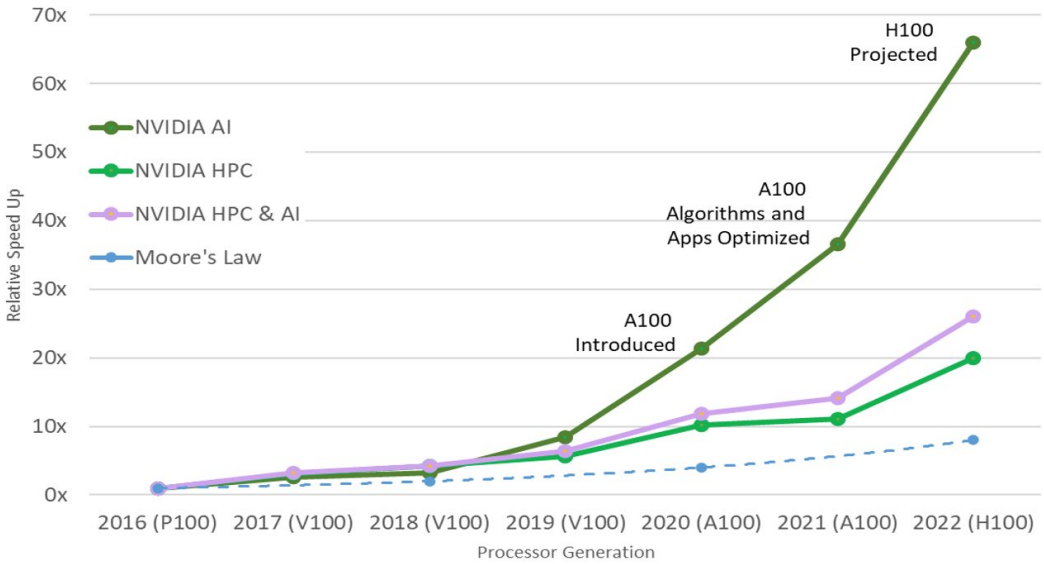


図 2.6.2 代表的な 9 つの HPC アプリと 2 つの AI テストケースによる、アプリケーション実行性能推移

※1 NVIDIA GPU アプリケーションの平均実行性能は、次のアプリケーションの幾何平均にて算出
HPC> AMBER [PME-Cellulose NVE]; Chroma [HMC-MGmedium]; GROMACS [ADH Dodec]; GROMACS [STMV]; MILC [Apex medium]; NAMDSTMV); QE [ASURF112-JR]; VASP [Si256-VJT-HSE06]; ICON [QUBICC 160km].
AI > TensorFlow [ResNet-50]; Pytorch [Bert Large])

<NVIDIAのエコシステム>

NVIDIAは、GPUによる高速並列演算を提唱し、過去25年間、GPUコンピューティングをリードしてきた。その中で生み出されたものは、GPUハードウェアに関連する技術だけでなく、プログラミングモデル、コンパイラ、ライブラリ、パフォーマンスツール、GPUに向けて高速並列化された多様で3000を超えるコミュニティアプリケーションや

商用アプリケーション群、AIフレームワーク、さまざまなAIアルゴリズム、13,000社以上のスタートアップパートナーを含むエコシステムにある。NVIDIAの開発環境ツールキットのダウンロード数は380万を超え、計算環境はオンプレミスからクラウドまで、多様な選択肢の中から目的に応じて選定しアプリケーションを実行することが可能である。

富岳NEXTに向けて理化学研究所（以降「理研」）とNVIDIAが協業することによって、Society 5.0を実現するアプリケーションを含む、これまでに理研が開発した多数の重要なアプリケーションが、このグローバルに展開された唯一無二のエコシステムの上で、学術分野や産業界にて幅広く使用されることが期待できる。

<これからのスーパーコンピュータ>

スーパーコンピュータの活用分野は、あらゆる方向に拡大しており、その流れは2030年に向けてさらに加速すると考える。特に、5つのワークロード（科学技術計算・AI・エッジとの融合・量子コンピュータとの融合・デジタルツイン）が要となり、Society 5.0の実現を後押ししていくだろう。

科学技術計算（以降「HPC」）は、これからのワークロードにおいてもスーパーコンピュータの基盤である。HPCとAIは相互補完関係にありHPCにAIの手法を用いることで、従来のシミュレーションの性能をけた違いに向上することが期待できる。また、観測データや実験データなど、エッジで獲得されるデータは、HPCにより活用されるようになるだろう。デジタルツインは、シミュレーションからの入力、AI代替モデル、そしてエッジからの観測データを使用して、現実世界と仮想世界の間でのリアルタイムなフィードバックループを可能にし、それを利用する研究機関や企業などは対話型に必要な情報を入力し即座に結果を受け取ることによって次に取るべきアクションをリアルタイムに決定していくことができるようになる。量子コンピュータは、今後10年間で実用レベルに到達するのは難しい可能性はあるものの、古典スーパーコンピュータによるエミュレーションや応用研究は、現在すでに世界中で競争が始まっている。

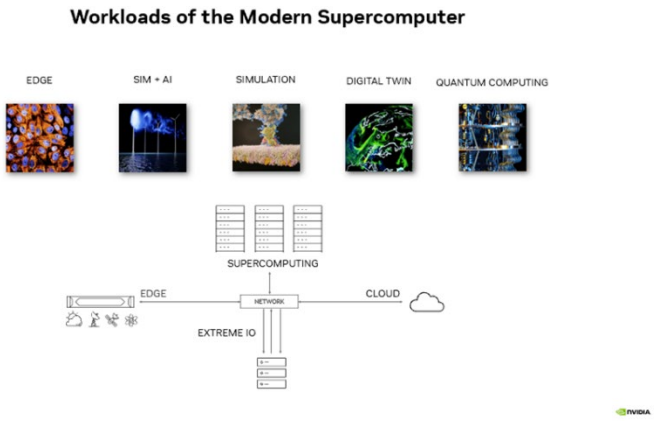


図 2.6.3 これからのスーパーコンピュータのワークロード

<電力性能向上へのコミットメント>

ムーアの法則の終焉や地球温暖化への対応において、アクセラレーテッドコンピューティングこそが性能を最大化しながらエネルギーコストを削減するための道筋であるとNVIDIAは考える。2022年11月に公表されたGreen 500リストにおいて、最も電力効率の高い30のシステムはすべてNVIDIAあるいはAMD社のGPUを搭載していることも実績として示している。電力性能向上へ向けての取り組みについては、次の報告書でさらに報告する。

2.7 アーキテクチャ調査研究サブグループ 6（HPE）

2.7.1 調査研究の概要

HPEでは、ポストエクサスケールのシステムアーキテクチャに対する複数のアプローチを検討している。

直近10年の後半 (2025年～2030年) には、気候変動などの地球規模の課題やスマートシティを含む社会全体の目標に向けた取り組みにおいて、世界が必要とする科学の問題を究明していくスピードを維持するために、HPCとAI主導のワークロードとワークフローの組み合わせが不可欠である。HPEは、これらの課題に対応するためにはインテリジェントなHPCシステムの活用が極めて重要になると考えており、ここで説明する複数のテクノロジに投資している。そうしたテクノロジはさまざまな組織や企業との連携を深めることによって、影響力を高められると見ている。

HPEは、さまざまなノードアーキテクチャに基づくコア処理技術の開発を、CPU/GPUの主要ベンダであるAMD、Intel、およびNVIDIAと連携して取り組んでいる。これらのベンダは、PCIeやCXLなどの業界標準の接続規格と比較して、ノード内の直接接続されたチップ内で優れたパフォーマンスを発揮する独自のローカルインターコネクト (xGMI、UPI、NVLink) の開発を続けている。HPEでは、次世代計算基盤の開発期間を通してこの取り組みが継続されることを期待している。HPEは、新たな代替プロセッサテクノロジの台頭を引き続き注視しているが、主要ベンダのこれまでの投資規模を考えると、ウェハースケール統合のような画期的なアプローチだけが対抗できると見ており、この開発期間中にウェハースケール統合に向けた開発努力がどこまで実を結ぶか注視している。このアプローチを推し進めれば、個別のプロセッサ間の第1レベルのインターコネクトを根本的に排除し、ウェハ上の各プロセッサを高帯域幅/低レイテンシ/省電力で接続できるようになる。HPE Slingshotイーサネットの機能は、ウェハースケールデバイスに直接統合でき、ウェハーレベルのプロセッサをスケールアウトすることができる。

HPEは、高度なプロセッサノードを結合してバランスのとれたシステムアーキテクチャ構築を実現するために、HPE Slingshotイーサネットのさらなる進化を目指す研究開発に投資している。これには、チップレットレベルのプロセッサとの統合の可能性の検証やノード間通信のためのシリコンフォトニクスの使用が含まれている。HPEは、同様の方法で日本のプロセッサ開発元とも協力している。HPEは、Slingshotによるエクサスケールの成功から得られた知識を活用して、ハイパフォーマンスなイーサネットの共通オープン標準を確立するために、広範な業界コンソーシアムのメンバとしても活動している。この取り組みは、次世代計算基盤プログラムの一環としてテクノロジを使用する際に、発生する可能性のあるリスクを軽減することにつながる。

HPEは、次の分野でのコラボレーションへの注力が、次世代計算基盤のアーキテクチャにもたらす影響にとって最も生産的であると考えている。

- HPE Slingshot イーサーネットインターコネクト、ならびに現在の開発方針の一例であるシステムオンチップ (SOC) アプローチから、コンピューティングノードあたりの処理速度を最大 50 倍にでき得る完全なウェハースケール統合を活用するための極めて画期的なアプローチにまで至る、処理技術との緊密な統合
- システム管理や HPC と AI 向けの生産性に優れたプログラミング環境を含むシステムソフトウェアスタック

次世代計算基盤の処理ノードについては現在、ローカルメモリ帯域幅に関して20TB/s達成を目標にしており、システムのバランスを保つためには、インターコネクト帯域幅の目標をその5%の範囲 (すなわち 1TB/s) に収めることを目指している。今後5年から10年の間には業界標準のイーサネットが800GB/sに達する見込みであるため、8～10本の密結合したSlingshotイーサーネットチャネルを各処理ノードで有効にすることを見据えている。ネットワー

クエンドポイントは、参照の局所性を維持するために、処理ノードのオンチップネットワーク上に分散される (現在、NICはマルチGPUノード上に分散されている)。この密度を実現するには、集積シリコンフォトニクスが必要になることが考えられ、処理デバイスやインターコネクトスイッチ用のデバイスと同一パッケージ化することも視野に入れている。

10年後に、どのリンクテクノロジが最大の価値を提供するかについては、まだ確実な見通しを立てることはできない。現在利用可能な最速のリンクテクノロジ (1600Gbps) が、その時点でも最も費用対効果が高いとは限らない。HPEの設計は、ノードごとに複数のエンドポイントとNICごとに複数のリンクを持たせ、高度に並列化されている。これは、各ノードで実行されているアプリケーションの優れた同時実行性を反映しており、多くのプロセスがネットワークトラフィックを生成していることを示している。ネットワーク設計におけるこのような柔軟性を利用して、必要なレベルのグローバルな帯域幅を費用対効果が高く提供するための最適化を図ることができる。

HPEでは、ベースプロセッサおよびインターコネクトテクノロジに加えて、次世代システムのソフトウェア要件を調査している。

HPEは、HPE Performance Cluster Manager(HPCM)とCray System Management(CSM)で従来のHPCシステム管理環境と現在のHPE製品で使用されているクラウド運用環境を組み合わせた新しい環境を構築する予定である。また、テクノロジパートナーとのコラボレーションの機会を最大限に活かすために、オープンソースを使用するアプローチを計画している。

プログラミング環境に関しては、HPEは従来の大規模なHPCアプリケーションの強化を図るために、HPE Cray Programming Environment (CPE) を進化させる取り組みを継続している。HPEは、AI/MLコードを「大規模に」実行し、多数のノードを使用したモデルトレーニングを推進するために、HPE Machine Learning Development Environment (MLDE) の開発も進めている。MLDEはオープンソースコードをベースとしているため、特に日本で開発された新しいプロセッサノードアーキテクチャのサポートに関して、コラボレーションの機会が明らかに広がっている。

2.8 まとめと今後の課題

アーキテクチャ調査研究グループでは、グループ構成員およびA0～A6のサブグループにより、次世代計算基盤のアーキテクチャに関する調査研究を実施した。サブグループA0では、本グループ全体を主導するために、半導体技術・パッケージング技術・アーキテクチャの技術動向の調査、ベンチマークプログラムの特徴解析によるワークロード分析、次世代計算基盤への要件や評価指標の整理、システムとしてあるべき姿の検討、他のサブグループ調査研究の方向付けと推進、内容のレビューやフィードバックを行った。サブグループA1～A６では、マイクロアーキテクチャ、ノードアーキテクチャ、システムアーキテクチャ、および関連するシステム・ソフトウェアやベンチマークによるアプリケーションの性能推定等の個別に設定する調査研究対象について、技術的な調査検討を行った。以上により、成果として、半導体やパッケージング技術動向を踏まえた、システム全体やその構成要素の技術的可能性や性能に関する調査結果が得られた。

今後の課題は、以下の通りである。

- マイクロアーキテクチャ、ノードアーキテクチャ、システムアーキテクチャ候補とその設計空間等の詳細化
- 提案アーキテクチャに対する性能・コスト等の定量的見積り。ベンチマークの性能推定
- 技術的可能性を考慮した開発ロードマップの検討
- プログラミング生産性などに関する評価。システムソフトウェア・ライブラリ調査研究グループおよびアプリケーション調査研究グループと連携した、アーキテクチャ評価

3. システムソフトウェア・ライブラリ調査研究グループ

3.1 調査研究の概要および方針

3.1.1 概要

システムソフトウェア・ライブラリ調査研究グループ（以下、システムソフトG）では、これまでのフラッグシップシステムや第二階層システムのソフトウェアの開発工数や利用状況、そして既に構築されているエコシステムとの親和性も踏まえ、ソフトウェア資産として何をベースに、国内でどこまでを開発すべきか、もしくは国際協調が必要なソフトウェアへ優先度をつけて明らかしつつ、今後のロードマップを策定することが必要である。また、次世代計算基盤の産業界を含む幅広い応用分野での活用を促すためにも、従来のシステム・ソフトウェアだけでなくデータ利活用の促進、機械学習技術と第一原理シミュレーション、さらには大規模リアルタイムデータ処理の高度な融合、従来の共用HPCシステムとは次元の異なる高セキュリティの担保、などを主要検討項目と見据えた今後の日本の次世代計算基盤として行うべきソフトウェア開発について調査研究を行う。

3.1.2 方針

次世代計算基盤に係る調査研究は、候補アーキテクチャを取り巻くソフトウェア・エコシステムの状況やその後の商業展開を見据えた調査が不可欠であり、網羅的なシステム・ソフトウェアの調査研究が必要である。このためシステムソフト・ライブラリ調査研究グループでは、コンパイラ・プログラミング、スケジューラ・ランタイム 、通信ライブラリ、I/O・ストレージ・ファイルシステム、数値ライブラリ、AIフレームワーク、OS・仮想化・クラウド連携、自動チューニング、セキュリティに関する技術調査に加えHPCにおけるシステム・ソフトウェアの利用環境調査を行うことのより、利用実態から優先順位をつけそれに基づき、国内で開発・移植・チューニングすべきものと国際協調によりソフトウェア開発を行うべきかの決定を行う。

より具体的には、以下の調査研究計画に従い、主に以下の3つの調査研究を行うことを決定した。（但し、調査研究の結果や情勢により今後追加の調査研究をする可能性があり以下に限定されるものではない。）

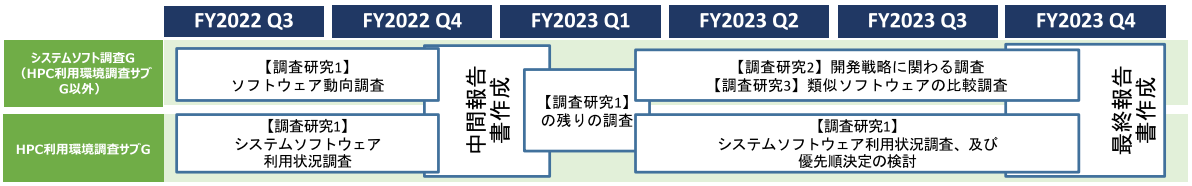


図 3.1.1 システムソフト G 調査研究計画

- 【調査研究 1】システム・ソフトウェアの動向調査と利用状況調査
- 【調査研究 2】システム・ソフトウェア開発戦略に関わる調査
- 【調査研究 3】類似ソフトウェアの比較調査

【調査項目1】では、移植性、生産性を加味して現在のシステム・ソフトウェアの動向と将来トレンドの調査し、また大規模システムにおけるシステム・ソフトウェアの利用状況の調査を行う。また、「富岳」開発や運用を振り返り次世代計算基盤に向けた技術的課題などの洗い出しも行う。【調査項目2】では、システム・ソフトウェア開発戦略に関わる調査、具体的には国内で独自開発もしくは海外連携により開発をするべきか、プロプライエタリソフトウェアとして開発もしくはオープンソースソフトウェアとして開発をすべきか、新規開発もしくは既存のソフトウェア・エコシステムを活用（移植、チューニング）した開発をすべきか、を決定するための調査を行う。例えば、近年、各チップベンダにお

いて自社のチップでのみ動作するだけでなく他者のチップ上でも動作する移植性の高いプログラミングフレームワークの開発が活発に行われているため、仮に国産のチップを次世代計算基盤で採用したとしてもその上で動作するプログラミングフレームワークは海外ベンダのフレームワークを活用することも考えられる。また、コンパイラにおいてはgccやLLVMに代表される通りオープンソースのコンパイラの利用が盛んに行われている。コンパイラに限らず、既存のソフトウェアエコシステムを活用することで、シフトウェアの開発コストを削減できるだけでなく、標準化されたソフトウェアを使用してアプリケーション開発をすることでアプリケーションの移植性も高まる。調査研究2では、システムソフト開発戦略を決定するためのソフトウェア・エコシステム等の調査を行う。【調査項目3】では、類似ソフトウェアの比較調査を行う。例えば、MPI実装のOpenMPI、MVAPICHはAPIが標準化されているため、どちらのMPI実装を採用したとしてもアプリケーションユーザへの影響は比較的少ない。また、一方が他方の代替ソフトウェアとしての役割も果たすため両方のソフトウェアを導入することでソフトウェアの欠陥があった場合には、ユーザの利便性を損なうことなく他方のシフトウェアへ切り替えが行えるためリスク軽減に役立つ。一方で、標準化されていないシステム・ソフトウェアを新規に導入する場合には、アプリケーションユーザは移植のためのコーディングが必要となる。例えば、Intel社のOneMKLや富士通社のSSL-IIは共に数値計算ライブラリであるが、数値計算ライブラリでは標準化が十分でないものもありその場合、移植の作業が必要となるため、富岳で採用されたArmベースのCPU以外のアーキテクチャを採用する場合には、その影響をある程度見積もる必要がある。このように、どのアーキテクチャ、またどのシステム・ソフトウェアを採用するかで生産性や移植性に影響するためあらかじめ類似シフトウェアに関する調査を行う必要がある。2022年度では、主に【調査項目1】について行なった。

3.2 各システム・ソフトウェア分野における中間調査報告

前節に述べた通り2022年度では主に【調査項目1】について行なった。この節では、2022年度までに行なった調査研究についてまとめる。その他、システム・ソフトウェアの現状と課題を洗い出すためにアプリケーショングループへ参画しているメンバに対しアンケート調査を行なった。そのアンケート調査の結果も当該の項においてまとめる。以下では、各システムソフトサブGの調査報告についてまとめる。

3.2.1 コンパイラ・プログラミングモデル

3.2.1.1 調査目的

次世代計算基盤は多様なアーキテクチャからなる大規模システムになることが想定される。そのような環境に対するアプリケーション開発者の準備状況や動向とともに多様なアーキテクチャに向けたさまざまなプログラミングフレームワークについて調査し、次世代計算基盤の活用のために必要な技術を洗い出す。

3.2.1.2 調査内容

本サブグループでは、（１）プログラミング言語に対するアプリケーション開発者のニーズやトレンド、（２）個々のアーキテクチャの性能を最大限に引き出すための、オープンソースコンパイラを含めた既存のコード生成技術やノード内におけるタスクベースのプログラミングモデル、（３）多様なアーキテクチャを想定した、パフォーマンスポータビリティを考慮したプログラミングフレームワーク、複数のアーキテクチャを統一的な枠組みでサポートするプログラミングフレームワーク、およびオフロードプログラミング、（４）これらをサポートするトレーサ・プロファイラ・デバッガなどの開発支援ツール、を調査対象として、国際的な動向を調査するとともに、我が国の次世代計算基盤に求められる機能の検討を行う。

3.2.1.3　調査結果

「富岳」の開発元である富士通とともに、「富岳」におけるコンパイラや開発ツールに対する振り返りを行った。「富岳」の当初の目的は達成できていることが確認できた。しかし、コンパイラやプロファイラの一部機能の使い方が分かりにくい、オープンソースソフトウェア(OSS)についてコミュニティへのフィードバックや新しいバージョンへの追従は不十分である、などの課題があり、「富岳NEXT」向けの開発ではそこに着目する必要がある。振り返り結果の抜粋を表 3.2.1に示す。

表 3.2.1「富岳」の振り返りによる成果と課題

開発目標・開発方針	成果	課題
高性能・ 高スケーラビリティの 実現	• CPU の特徴を活かした機能をコンパイラに実装し、高性能・高スケーラビリティを実現した	−
最新言語規格/ 業界標準仕様の サポート	• 各言語規格について、設計当初の最新規格をサポートした • コンパイラは、OSSであるClang/LLVMをベースとしたモードを導入したことにより、「京」のコンパイラより業界標準のGNU互換などが向上した	• コンパイラでは、ベースの Clang/LLVM に対して富士通独自の修正を加えたことにより、Clang/LLVM から GNU 互換が損なわれたところがある
使い易さの追求	• 「京」と同様の汎用的な開発環境を提供した • CPU マイクロアーキテクチャ分析を行いやすいプロファイラを提供した	• C/C++コンパイラで 2 つのモードを利用者に使い分けてもらうのは難しい • プロファイラの CPU マイクロアーキテクチャ分析機能は初めて使う人にとって分かりにくい
スーパーコンピュータ 「京」のプログラム 資産の継承	• CPU アーキテクチャは「京」から変わったがプログラム資産は性能も含めて互換性を維持した	―
オープンソース(OSS)の採用による開発の 迅速化・低コスト化	• 複数の OSS の採用により開発の迅速化・低コスト化を実現した	• OSS コミュニティへのフィードバックや OSS の新しいバージョンへの追従は不十分である
コデザイン・基本設計	• アプリケーションとのコデザインにより重点課題の目標性能を達成した	• コンパイラの最適化がコデザインのターゲット・アプリケーションのソースコード向けに偏ってしまう傾向がある
対外アピール・ エコシステム	• SVE 活用のコンパイラ技術を世界にアピールした • 「京」と比較するとソフトウェアのエコシステムが拡充された	• OSS コミュニティや他社との協業の観点でのエコシステム拡充は不十分である

プログラミング言語に対するアプリケーション開発者のニーズやトレンドを把握するために、アプリケーション開発者に対して「使用言語」「アクセラレータでの開発経験」等のアンケートを行った。回答者数は30である。下に示すように、使用言語では、Fortranが最も多く、次いでPython、C、C++となった。ただし、Pythonのみを使用

しているという回答は1名のみであった。他の回答者はFortran、C、もしくはC++をPythonと併用するか、Fortran、C、もしくはC++のみを使用していた。アクセラレータに関するプログラミングについては半数以上が経験がない、もしくはほぼないと回答した。開発経験者のうち6名はOpenACCを、2名はOpenMPを用いてGPU向けの開発を行っていた。さらに、現在、アクセラレータでの開発経験がないと答えた開発者の6割が、将来システムにアクセラレータが導入された場合、アプリケーションの拡張のために、directive base でのプログラミングがサポートされることが望ましいと回答した。

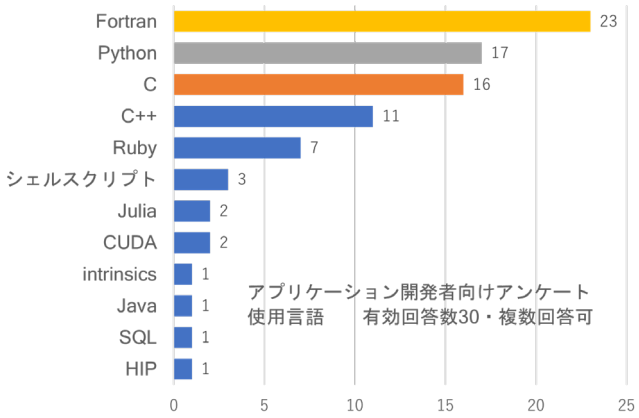


図 3.2.1

Performance Portable なプログラミングモデルについては、使用経験者は少数であったが、開発コストによっては今後使ってみたいという意見が多かった（図 3.2.2）。経験者からは現状ではバグが多く、アプリケーションによっては性能がでないとの指摘もあった。

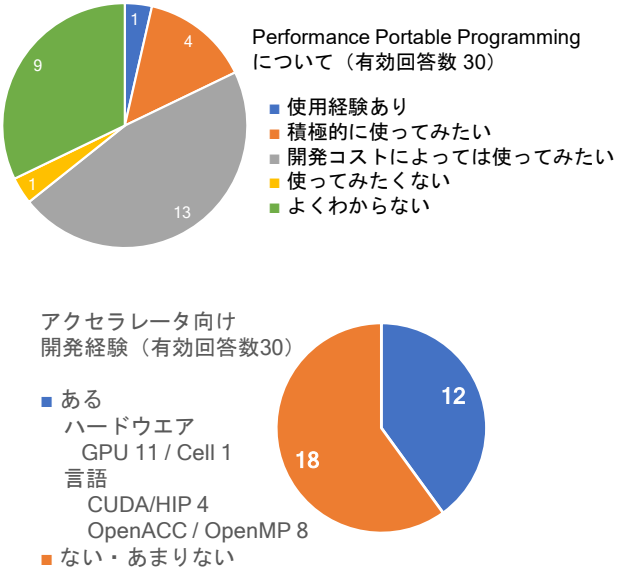


図 3.2.2

デバッグツール（表 3.2.2右）については、gdb、TotalView、objdump やコンパイル時の静的な誤りチェックの使用者が多かった。プロファイラ（表 3.2.2左）については、ベンダが提供している性能解析ツールが主であったが、gprofの使用者もあった。ベンダ制ツールを使用していると答えた10名のうち9名は富士通

製のプロファイラを使用していた。富士通製のプロファイラのうちでもCPU性能解析レポートは、エクセル形式でキャッシュミスや命令実行率などCPU性能に関する多様な情報を出力するツールである。多くの情報が1枚のエクセルシートにまとまっており解析が容易である一方で、csvのプロファイルデータをユーザのPCにダウンロードした上で、エクセルマクロを実行するという手間が必要になる。

表 3.2.2

プロファイラの使用状況（複数回答可）	
ベンダ制プロファイラ	10
gprof	2
printf	1
Linux perf	1

デバッガの使用状況（複数回答可）	
gdb	9
コンパイラオプション	3
Valgrind	2
ベンダ制デバッグツール	1
printf	1
objdump	1

ワークフローアプリケーションに対する調査では、12名の開発者から利用・利用予定との回答があった。気象、素粒子、生命科学、社会科学など幅広い分野で利用されていた。多くは 1～12ノードの小規模なタスクを数百から数千程度実行するとしていたが、1000ノードのタスクを逐次的に実行するとの回答もあった。タスクの開発やマネージメントは、OpenMP、MPI、Python、Javaなどを用いてそれぞれ実装されていた。

コンパイラに関して特筆すべき大きな流れは、多くのベンダがプロプライエタリのコンパイラからオープンソースのコンパイラやオープンソースベースのコンパイラへと移行しつつある、あるいは移行している、という点である。表3.2.2に現状のHPCベンダのコンパイラの提供状況を示す。なお、ArmについてはHPC向けのコンパイラのみを考慮し、組み込み向けのコンパイラは考慮していない。

表 3.2.3

	富士通	インテル	AMD	NVIDIA	HPE	Arm
オープンソース・ オープンソースベース	○	○	○	○	○	○
プロプライエタリ	○	○		○	○	

オープンソースのコンパイラの利点として、特定のアーキテクチャやソフトウェア向けのコンパイル・最適化を計算機センタ、研究機関、およびユーザが追加できることが挙げられる。国内におけるそのような取り組みとして、理研のLLVMに対するA64FX向けソフトウェアパイプライニングの機能の実装、東大と理研による再構成可能型のアクセラレータ向けコンパイラ[1]などがある。後者では、最適化パスにおいて、従来手法に加えて独自に遺伝的アルゴリズムを取り入れている。

GPU等のアクセラレータに向けたプログラミングモデルとしては、CUDAやHIPなどの特定のアーキテクチャに向けた言語、クロスプラットフォームAPIであるOpenCL、より抽象的なプログラミングが可能なOpenACCなどが用いられている。さらに、C++をベースとしてさまざまなハードウェアに向けたプログラミングが単一的に行えるPerformance Portableなプログラミングモデルに関する研究開発も行われている。とくに、世界トップクラスのCPU+GPUシステムである米Frontier や米 Aurora （開発中）においては、OpenMP、OpenACC、OpenCLなどの既存のプログラミングに加えて、SYCL やそのIntel-GPU向け拡張であるDPC++やAMD-GPU向け拡張であるhipSYCL、Kokkos なども用いられている。C++23以降の言語標準にアクセラレータ向けの新標準が提案される動きもある。これらのPerformance Portableなプログラミングは、現時点では

国内のアプリケーション開発者にはあまり用いられていないが、システムの相互利用を含む国際共同研究の増加を踏まえ、既存のアクセラレータプログラミングと同様に、将来システムではより重要な位置を占めると考えられる。ただし、アプリケーション開発者へのアンケートにもあったように、現状のPerformance Portableなプログラミングは、常に高い性能が出るとは限らず、今後も調査が必要である。

図 3.2.3に、HPCにおけるプログラミングモデルの外観を示す。なお、赤太字は「富岳」でサポートされているコンパイラ、ライブラリ等を示す。

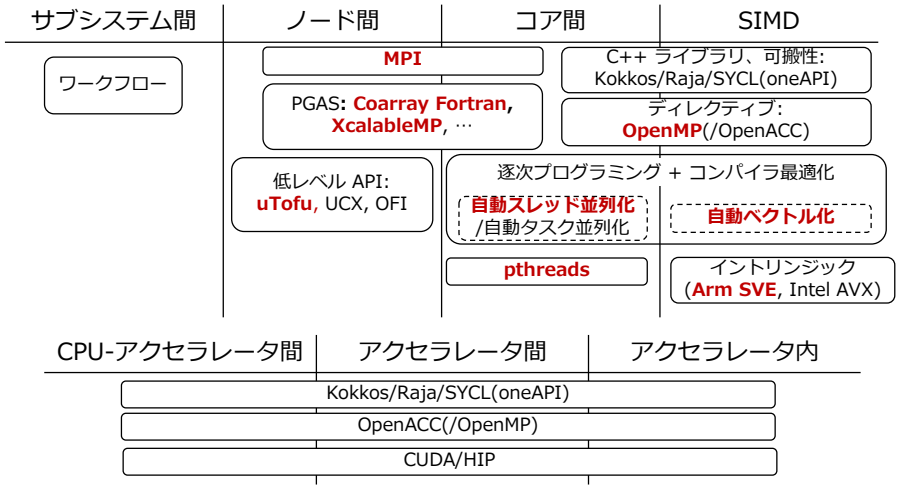


図 3.2.3

ワークフロープログラミングについては、ワークフロー開発実行環境の多くは目的別にオープンソースで開発されており[2]、スタンダードは存在しない状況である。共通化に向けた取り組みも複数の分野で独立に行われており、バイオ分野のcommon workflow language (CWL)、人工知能に関するAI workflow、HPC関係者によるFAIR Computational workflow [3] などがある。国内においては、上述の開発者アンケートから強いて広く用いられているものをあげるとすれば、OpenMP、MPIおよびPythonなどの既存のプログラミング言語・API・ライブラリの使用および拡張である。海外の比較的大規模なプロジェクトやフレームワークとしては、欧州eFlows4HPCプロジェクトで用いられているCOMPSs/PyCOMPSs[4]、米テネシー大学を中心に開発されているPARSECがある[5]。

次にデバッガと性能解析ツールの動向について述べる。国内外の計算機センタで広く使われる統合デバッガとしては、DDTおよびTotalViewがある。いずれも商用ソフトウェアであり、GUIベースでアクセラレータや並列環境でのデバッグもサポートする。計算コアでのデバッグやメモリのデバッグなど特定の用途に広く使われるものとしては、gdb, Valgrind, STAT, ReMPI等があげられる。性能解析ツールは、商用非商用問わずさまざまな目的で多くのツールが存在する。最も重要なのは、ハードウェア上に実装されたハードウェアカウンタおよびそのAPIであるPerformance API (PAPI) である。アプリケーション開発者アンケートでも使用者が多かったベンダが提供している性能解析ツールは、多くの場合、ハードウェアカウンタの情報に基づいた性能解析データを提供している。PAPI、通信、アクセラレータなどの統合的な解析ツールとして、HPCToolkit, MAP, TAU, Score-P, Vampirなどがある。I/Oを対象とした解析ツールとしてはDarshanが広く用いられる。

3.2.1.4　来年度計画

来年度については以下を調査する：

- アプリケーション開発者向けのアンケートなどの調査結果を踏まえ、Fortran で記述されたアプリケーションのアクセラレータでのサポート方法についての調査を行う
- Python の HPC 向け最適化について調査を行う
- ベンチマーク・サブワーキンググループより提供されるベンチマークを用いて、オープンソースベースコンパイラとプロプライエタリコンパイラの比較を行うとともに、必要性の高いコンパイラ最適化機能について調査する
- 同ベンチマークを用いて Performance Portable なプログラミングモデルに対する性能評価。実行性能のほか、移植コストの調査を行う
- ワークフロープログラミングについて、動向調査を継続して行う
- デバッガや性能解析ツールについて、動向調査を継続して行う

[1]小島拓也, et.al., “HPC向けRIKEN CGRAのためのコンパイル環境整備と予備評価,” 研究報告システム・アーキテクチャ（ARC）, vol. 2022-ARC-247, no. 23, pp.1-6, 2022.
[2] https://s.apache.org/existing-workflow-systems
[3] https://arxiv.org/pdf/2110.02168.pdf
[4] https://eflows4hpc.eu/
[5] https://icl.utk.edu/parsec/

3.2.2 スケジューラ・ランタイム

3.2.2.1 調査目的

次世代計算基盤は多種多様かつ膨大な量の計算資源から構成される超大規模システムになることが想定される。一方で、ワークフロー実行やコンテナ仮想化技術の普及により、次世代計算基盤で実行されるアプリケーションや実行方法も現在よりさらに多様化する。時間的特性、空間的特性、資源分離レベルの異なる多数のジョブを、多種多様な計算資源に適切に割当てることは、それ自体が挑戦的な技術的課題である。特に、現在および将来において我が国で開発されるシステムやアプリケーションの効率的なスケジューリング技術およびその実行支援技術(ランタイム)は、我が国が独自に検討しなければならない。それらの技術の研究開発を進めるために、国際協調の下で研究開発が進んでいるジョブスケジューリング技術の動向把握と、我が国が独自に追加検討すべき機能要件の調査を行う。同様に、ワークフロー管理ツールやモニタリングツール、および耐障害性機能を実現する実行支援技術についても調査する。

3.2.2.2 調査内容

ジョブスケジューラとその関連技術について、情報基盤センタごとに現在の利用状況および今後の運用方針に関して調査して要求要件の整理を行うほか、アプリケーションの効率的なスケジューリング技術およびその実行支援技術に関して国際協調の下で研究開発が進んでいるジョブスケジューリング技術の動向把握と我が国が独自に追加検討すべき機能要件の調査を行う。

3.2.2.3 調査結果

ジョブスケジューラの利用状況調査

現在一般的に利用されているジョブスケジューラには、いくつかの系統がある。長期間にわたる運用が期待される次世代計算基盤では、その期間全体にわたって、運用に合わせたジョブスケジューラの保守と改修が求められる。このために、今後も国産ジョブスケジューラの開発を維持していくか、あるいは国際標準ジョブスケジューラに必

要な機能拡張する場合にはそれが広く使われており将来にわたって保守管理されていく見込みがなければならない。

このような背景から、実運用システムで広く使われている有力なジョブスケジューラを把握するために、TOP 500 Listにおける上位100システムでの採用状況、および各情報基盤センタの採用状況を調査した。TOP 500 Listには、利用されているジョブスケジューラの情報は記載されていないため、採用状況を調査するためには、各システムで利用しているジョブスケジューラを個別に調査する必要がある。本調査の目的は各ジョブスケジューラを利用するシステム数を正確に知ることではなく、国際的な採用状況の趨勢を知ることにある。このため2022年度には各システムに関する公開情報からジョブスケジューラを推定する作業を行い、国際的に有力なジョブスケジューラの特定と、各ジョブスケジューラの採用状況を調査した。具体的には、同様の調査結果が以下のアドレスにてすでに公開されていたため、そのデータに基づいて最新のTOP 500 Listの上位100システムについて集計し、運用ジョブスケジューラが不明とされていたシステムに関しても、利用者向けに公開されている各システムの資料を参照する等の方法により可能な限り調査した。

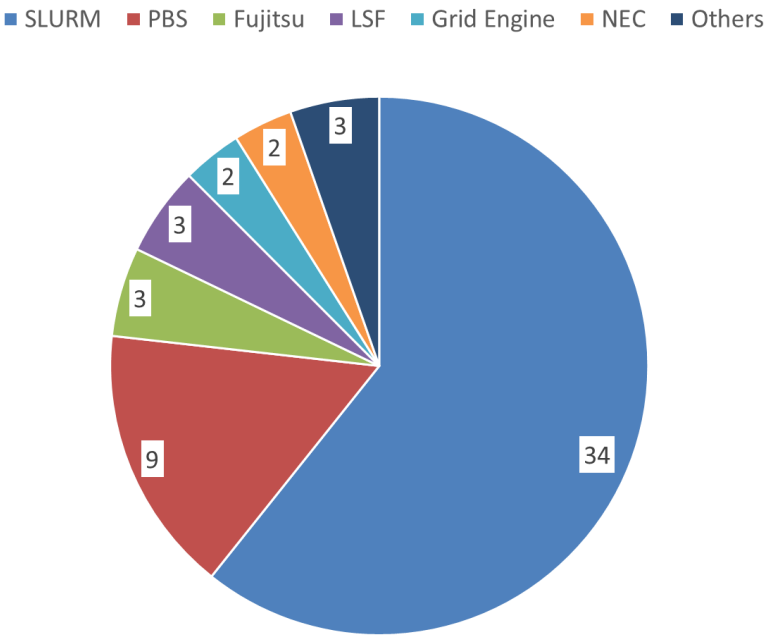


図 3.2.4 2022 年 11 月の TOP 500 List の上位 100 システムにおけるジョブスケジューラのシェア

https://speakerdeck.com/porcaro33/top500-scheduler

本調査の結果、上位100システムのうち56システムのジョブスケジューラを推定することができた。2022年11月の上位100システムに対する調査結果を図 3.2.4に示す。この集計結果から、国際的な採用状況として、大規模システムの運用にSLURMを採用する事例が多いことが分かる。ただし、ジョブスケジューラを独自に開発・提供しているベンダ(IBM、富士通、NEC)が納入したシステムにおいては、そのベンダのジョブスケジューラが採用されている例も多い。

国内でも同様の調査を行ったところ、多くの情報基盤センタの運用システムにおいてベンダのジョブスケジューラが採用されている一方でSLURMの採用事例は少なく、国際的な採用状況とは様相が異なっていることが明らかになった。ジョブスケジューラは各システムの運用方針と密接に関係しており、前述の通り、運用期間全体にわたって長期間の保守管理も必要になる。このため、実運用システムのジョブスケジューラの選択においては、ベンダ

にとっての保守管理のしやすさも重要な判断基準となっていると考えられる。複数の国内ベンダに対する聞き取り調査でも、同様の意見を聞くことができた。

ジョブスケジューリングの課題整理

実運用システムにおけるジョブスケジューラの採用状況に関する趨勢は図 3.2.4の通りとなったが、次世代計算基盤で効率的なジョブスケジューリングを実現するためには様々な技術的困難が予想されるため、これまでの運用実績や保守管理のしやすさといった運用コストだけでは、今後研究開発すべきジョブスケジューリング技術の課題を明確にすることはできない。このため、文献調査やベンダとのミーティングを行い、今後予想されるジョブスケジューリングの技術的課題についても調査を進めた。

文献[1]では、従来とは異なる多様で不均質なワークフローを実行するエクサスケールシステムで想定される技術的課題として、協調動作する複数のジョブを十分なスループットでスケジューリングすること(throughput and co-scheduling challenges)が挙げられ、その対応のために階層的なジョブスケジューラを提案している。また、ジョブ間の協調や通信を記述するためのインターフェースも定義しており、性質の異なる多数のジョブをシステム上で連携して動作させることを強く意識している。文献[2]では、集中型ジョブスケジューラであるSLURMを分散化するとともに、機械学習に基づいてジョブの実行時間を予測することでスケジューリングを効率化している。文献[3]では、通常のジョブに加えてオンデマンドジョブや並列数可変のジョブ(malleable job)が実行されるシステムにおける効率的なジョブスケジューリングを議論している。いずれも、システム構成とそこで実行されるアプリケーションやワークフローの双方が大規模化、多様化、複雑化することを前提としており、その効率的なジョブスケジューリングは依然として確立されていない重要な検討課題であることが分かる。

文献調査に加えて富士通株式会社の技術者とのミーティングを複数回行い、京や富岳のジョブスケジューラ開発を振り返るとともに今後に向けて検討すべき課題についても議論した。京のジョブスケジューラ開発においては、Tofuネットワーク中の連続したトーラス単位でジョブを割当てつつ高いシステム利用率を達成し、8万ノード規模の京で安定した運用を実現した。一方、富岳開発当時のオープンソースソフトウェアのジョブスケジューラではその規模での運用実績はなかった。このため、富岳においても京で培ったソフトウェア技術を継続して利用し、さらに発展させることで30万ノード規模の並列実行にも耐えられるソフトウェア基盤を構築した。具体的には、ジョブスケジューラの管理ノードを階層化して負荷分散・並列化することで、システムの大規模化に対応した。管理ノードで情報を集約し、整合性を保ったままジョブを管理・制御することができる。ネットワークトポロジを意識して計算ノードを割当てるだけではなく、計算ノード内の資源をNUMA構成を考慮して割当てることも実現している。このように、ジョブスケジューリング技術はシステムアーキテクチャを強く意識して検討していく必要があり、次世代計算基盤においても我が国が独自のシステムアーキテクチャを研究開発していく際には、その独自性を考慮して効率的利用に資する機能を追加していく必要がある。ただし、その実装において従来通り国産ジョブスケジューラを保守管理していくべきか、あるいは国際標準のジョブスケジューラに機能追加をしていくべきか、という点に関してはシステムアーキテクチャにも依存する問題であり、今後のシステム設計の進捗に合わせて引き続き検討していく必要がある。

また、富岳のジョブスケジューラで対応が不十分な点として、「外部システムとの連携機構」が示された。次世代計算基盤は、ユーザニーズの多様化や利用分野の拡大・変化に対応するため、「フラッグシップシステム」および国内の主要な計算基盤、データ基盤、ネットワークが一体的に運用され、総体として持続的に機能する基盤となることが望ましい [4]。ジョブスケジューラ開発においても、例えば外部システムからジョブの管理や結果の取得を行うためのAPIを提供するなど、システム外部の計算基盤・データ基盤・ネットワークと連携する機構を検討する必要がある。

ワークフロー管理ツール

ジョブスケジューラの調査に加えて、ワークフロー管理ツールについても動向調査を開始した。いくつかの文献調査に加えて、理化学研究所で開発されているワークフロー管理ツールであるWHEEL [5]について開発グループから聞き取り調査を行った。

モニタリングツール

効果的なジョブスケジューリングを実現するためには、計算資源の状態をリアルタイムかつ詳細に監視することが不可欠である。そこで、HPCシステムで用いられているモニタリングツールの文献調査も開始した。本年度は特にストレージに着目し、業界標準のI/Oプロファイリングツールであるdarshan [6] について調査した。

Darshanは米国アルゴンヌ国立研究所において開発されているI/Oプロファイリングツールであり、アプリケーションのコンパイル時または実行時にI/O処理に関係する関数をフックすることにより、ユーザから透過的にI/O処理の性能情報を収集する。具体的にはアプリケーションがアクセスしたファイルごとに、ファイル名、I/O操作の種別および数、アクセスサイズ、ストライド長、実行時間などの性能情報を収集する。フックする対象はPOSIX I/Oをはじめとして、MPI-IO、HDF5、Parallel NetCDFなどのより高位のI/Oライブラリにも対応している。また、Darshanは拡張可能に設計されており、MPI通信の性能情報や、相互結合網のパフォーマンスカウンタ値を収集するプラグインが提供されている [7]。

Darshanの特徴として、軽量であることがあげられる。一般的にプロファイラはオーバーヘッドを発生させるため、性能分析作業を実施するときのみアプリケーションに適用される。一方、Darshanはオーバーヘッドを軽減するために工夫がなされており、アプリケーションの性能に与える影響が非常に小さい。そのため、米国のALCF、OLCF、NERSCなどの計算機センタでは、実運用システムにおいてDarshanがデフォルトで有効化されており、全てのジョブの性能情報を収集している。

このように実運用環境においてアプリケーションの性能情報を常に計測することを継続的プロファイリング (continuous profiling) と呼び、クラウド分野では広く普及しつつある。米国Google社が自社のデータセンタにおける継続的プロファイリングの利用を明らかにして以降 [8,9]、継続的プロファイリングを実現するための様々なサービスやソフトウェアが開発・提供されている。このように蓄積された性能情報は、効率的なシステムの運用やアプリケーションの最適化に有用であるほか、システムの障害や性能低下などの異常の早期発見や原因究明に役立てられることが期待される。

3.2.2.4　来年度計画

2023年度の計画として、システム大規模化に向けたジョブスケジューラの負荷分散に関してさらに調査を進める。また、アプリケーション利用形態の調査に基づき、受付可能なジョブ数、同時実行数など、ジョブスケジューラに要求される機能と性能要件をまとめる。さらには、従来のジョブスケジューリングではあまり重視されて来なかった以下の項目について、調査する計画である。

- コンテナ仮想化技術 (Singularity や Shifter 等)やそのオーケストレーション技術 (Kubernetes 等)と、ジョブスケジューラとの相互運用
- 時間的特性、空間的特性、資源分離レベルの異なる多数のジョブを連携させる複雑なワークフローへの対応
- システム外部の計算基盤・データ基盤・ネットワークと連携する外部連携
- 性能分析や耐障害性などを実現する実行支援ツール

参考文献
[1] Ahn et al., “Flux: Overcoming Scheduling Challenges for Exascale Workflows,” Future Generation Computer Systems, Vol. 110, pp. 202-213, 2022.
[2] Dai et al., “Toward Scalable Resource Management for Supercomputers,” International Conference for High Performance Computing, Networking, Storage, and Analysis (SC), 2022.
[3] Fan et al. "Hybrid Workload Scheduling on HPC Systems", International Parallel & Distributed Processing Symposium (IPDPS), 2022.
[4] 次世代計算基盤に係る政策・技術動向 次世代計算基盤検討部会 中間取りまとめ 概要 (https://www.mext.go.jp/content/20210913-mxt_jyohoka01-000017979_02.pdf)
[5] https://github.com/RIKEN-RCCS/WHEEL
[6] Carns et al., “Understanding and improving computational science storage access through continuous characterization,” ACM Transactions on Storage, Vol. 7, No. 8, pp. 1-26, 2011.
[7] Liu et al., “Characterization and Identification of HPC Applications at Leadership Computing Facility,” 34th International Conference on Supercomputing (ICS’20), 2020.
[8] Ren et al., “Google-Wide Profiling: A Continuous Profiling Infrastructure for Data Centers,” IEEE Micro, 2010.
[9] Kanev et al., “Profiling a warehouse-scale computer,” 42nd Annual International Symposium on Computer Architecture (ISC’15), 2015.

3.2.3 通信ライブラリ

3.2.3.1 調査目的

本通信サブグループでは、広範なSDGs、Society 5.0の実現に向けた課題解決のためのプラットフォームとなる次世代大規模高速計算機において求められる通信ライブラリの機能と性能の調査を行っている。この通信ライブラリに求められる要件として、以下が挙げられる。

- シミュレーション、AI、デジタルツイン、ワークフロー実行などの多様なソフトウェアに対し、電力制約の下でデータ移動を高度化、効率化するため、特に強スケーリングやデータフローのためのソフトウェア技術
- 既存エコシステムとの親和性や国際協調を踏まえたシステム・ソフトウェア整備
- 幅広いアプリケーションについて、アーキテクチャ、アルゴリズムとのコデザイン
- 機械学習と第一原理シミュレーションとリアルタイムデータ処理の融合
- アプリケーションによっては従来の HPC システムとは次元の異なる高セキュリティの担保

3.2.3.2 調査内容

本通信サブグループでは、次世代スーパーコンピュータへの採用が想定されるネットワーク技術や様々なアプリケーションを考慮し、スケーラブルな並列計算の実現に向けて必要となる新しい通信ライブラリのインターフェースや実装技術を調査する。特に、今後登場する新しいネットワーク技術の活用、およびその活用手段のアプリケーションレベルでの表現、という視点で調査する。

3.2.3.3 調査結果

本通信サブグループの本年度の主な調査結果は以下のとおりである。

● 通信ハードウェア動向調査

現在、多くのスーパーコンピュータで利用されているインターコネクト技術である InfiniBand について、NVIDIA 社に今後の動向を聞いた。現時点で確定しているのは 2023 年に登場予定の XDR 規格、および BlueField-3 である。このうち XDR は、現在の NDR 規格の 2 倍の 800Gbps の通信速度により、今後の並列計算における通信データの大規模化に対応するものである。また、XDR 対応のスイッチは最大ポート数が 144 ポートとなり、大規模並列計算機の大量の計算ノード群間を、より少ない通信ホップ数で接続可能とするため、通信遅延時間の短縮にも貢献すると期待できる。一方 BlueField-3 は、同社が DPU と呼んでいる、ネットワークアダプタ上に搭載する汎用プロセッサである。同様の技術は、Intel 社等、他のベンダのネットワーク装置でも、SmartNIC などの名前で搭載が進められている。従来、ホスト CPU が担当していた、通信プロトコル処理や分散ストレージ管理などの通信関連の処理を DPU にオフロードすることにより、CPU を計算に専念させて計算効率の向上を図ることが出来るほか、計算と通信処理を並行して進めることにより、特に並列アプリケーションの強スケーリングを実現するうえで重要となる通信時間の隠蔽に利用できる（図 3.2.5）。また、DPU 上での暗号化、複合処理についてもインターフェースが整備されており、ノード間通信のセキュリティ高度化が求められる場合への対応が可能となる。BlueField-3 は、従来の BlueField-2 の 5 倍のメモリバンド幅と 2 倍の計算性能を有する予定であり、前述の DPU に対する期待に対して、さらに高いレベルで応える技術になると予想される。

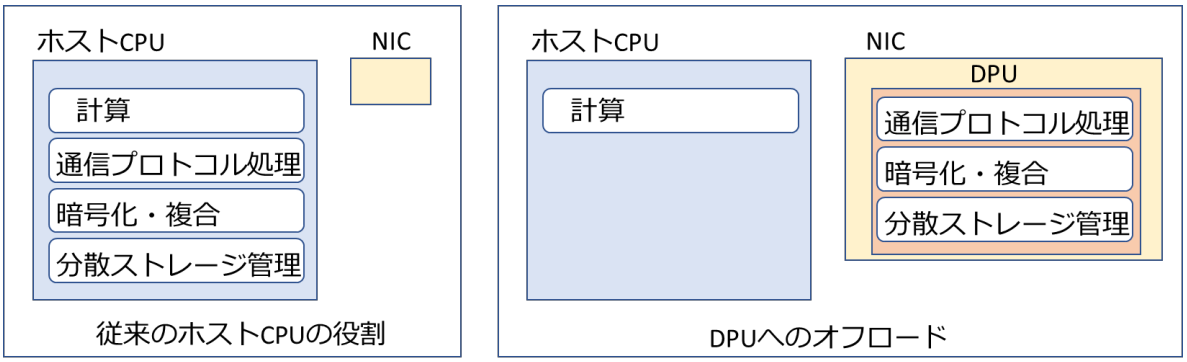


図 3.2.5 DPU による効果

一方、次世代計算基盤の導入が予定される 2028 年前後のインターコネクトアーキテクチャに関する予想が、「次世代計算基盤に係る調査研究事業」でアーキテクチャを担当する富士通社により、2023 年 3 月の情報処理学会ハイパフォーマンス研究会で報告された。これによると、計算ノード間を接続する伝送メディアの通信速度としては、400Gbps～800Gbps が予想されるものの、価格や安定性を考慮すると、400Gbps を想定するのが現実的である、との指摘があった。一方、ネットワークトポロジについて、現在のスーパーコンピュータで用いられている、多次元 Torus、Fat Tree、Dragonfly が検討された。その結果、ノード数が 32,768 以下であれば Dragonfly、これを超えるノード数であれば多次元 Torus が推奨されるとの報告があった。一方 Fat Tree については、次世代計算基盤で想定されるノード数の場合、高額なディレクタスイッチを使用した多段構成が必要となるため、主に価格面で妥当ではないとの指摘があった。ただ、研究会での質疑で、Fat Tree について Full Bisection Bandwidth ではない構成とすることで、選択肢の一つとして残るのではないか、との議論があった。

今年度の調査のまとめとして、2028 年あたりのインターコネクト技術について、通信帯域幅は 400Gbps～800Gbps が予想され、通信速度が向上するものの、通信遅延については、今のところ大幅な低減を期待できる技術が存在しないことが分かった。そのため、強スケーリングの実現に向けて、通信隠蔽が重要となることが改めて確認できた。この通信隠蔽を実現する技術の候補として NVIDIA DPU に代表される SmartNIC が挙げられる。一方、既存の通信隠蔽技術としては、スーパーコンピュータ「富

岳」でも採用されているアシスタントコアの他、高機能スイッチへのオフロード、ホスト CPU の余剰コアを用いた通信スレッド、等があり、これらと SmartNIC について、性能や利便性などの観点での比較が必要である。

- 通信ソフトウェア動向調査

  今回の調査では、通信ソフトウェアの調査対象を、大きく、低レベル通信レイヤ、高レベル通信レイヤ、アプリケーションの 3 つのカテゴリに分ける。低レベル通信レイヤは通信ハードウェアに近い通信ソフトウェアで主に通信デバイスの違いを隠蔽することを目的としており、UCX (Unified Communication X)、Libfabric などが含まれる。高レベル通信レイヤはアプリケーションに近い通信ソフトウェアで主にアプリケーション内の通信を表現する API の提供と、その効率的な実装を担当し、各種 MPI ライブラリなどが含まれる。アプリケーションは、エンドユーザが利用するアプリケーションソフトウェアの他に、数値計算ライブラリなどの各種ライブラリソフトウェアや、分散ストレージなど、エンドユーザが暗黙的に利用するユーティリティソフトウェアも、通信の視点からはアプリケーションとして扱う。このうち今年度は、主に低レベル通信レイヤと高レベル通信レイヤの動向を調査した。

  低レベル通信レイヤとして、今年度は主に UCX を調査した。UCX は UCF (Unified Communication Framework) のプロジェクトの一つであり、主要な MPI ライブラリ、および OpenSHMEM のような高レベル通信レイヤの下位レイヤとして採用されている。そのため、今後も事実上の国際的な標準通信レイヤとして普及が進むと想定される。今後の開発について、UCX は国際会議 SC22 の BoF での発表によると、DPU 向けインターフェースの開発と、CXL 関連の機能追加が予定されていることが報告されている。一方、今年度は未調査だが、同様の低レベル通信レイヤとして Libfabric がある。これは OFA (OpenFabrics Alliance) のサブグループの一つである OFIWG (Open Fabrics Interfaces Working Group) が開発している低レベル通信レイヤであり、やはり主要な MPI ライブラリの下位レイヤとして採用されている。また、Libfabrics の下位レイヤとして UCX を選択することも可能となっている。

  高レベル通信レイヤとして、今年度は MPI を調査した。まず MPI の規格を策定する MPI Forum では、現在主に MPI 4.1 規格の文書化と、MPI 5.0 規格の検討が進められている。このうち MPI 5.0 で予定されている機能追加として、計算機のトポロジ情報を活用した効率化支援機能、スレッド並列と組み合わせたハイブリッド並列計算の支援機能、デバッグ支援機能、などが提案されている。また、MPI ライブラリについては、オープンソースのもののうち特に広く利用されている Open MPI、MPICH、MVAPICH を調査した。Open MPI では、プロセス起動ツール PRRTE の導入や ULFM (User Level Failure Mitigation) の実装などが予定されている。MPICH では、MPI とスレッド並列を組み合わせたハイブリッド並列計算の支援機能として MPIX_Stream や Virtual Channel Interface を提案している。MVAPICH2 では、DPU による集団通信オフロードが進められている。

  アプリケーションについては、今年度のハードウェアおよびソフトウェアに関する調査結果をもとに、来年度、関連グループへのインタビューを実施する予定である。

  今年度の調査のまとめとして、低レベル通信レイヤについては、現時点で UCX と Libfabric が事実上の国際標準として使われており、今後も同様の状況になると考えられるため、次世代計算基盤では、これらのうち少なくともいずれか片方のレイヤを提供する必要がある。どちらを優先すべきかについては、来年度以降に調査する。高レベル通信レイヤについても、現時点では MPI が国際標準であり、次世代計算基盤でも不可欠である。MPI ライブラリの実装について、利便性の観点からは Open MPI、MPICH、MVAPICH のいずれかがそのまま動作することが望ましいが、次世代計算基盤に特有の機能や性能を発揮させるための API が必要となった場合、その実現方法については来年度以降に調査、検討する。

- アプリケーション開発者に対するアンケート

  通信ライブラリを使用しているアプリケーション開発者に対し、現在の通信ライブラリに対する要望についてアンケートを取ったところ、主に以下の回答が得られた。

  - 非同期通信による通信隠蔽、ランクマップ最適化、などを簡単に行えるインターフェース
  - 業界の標準的な通信インターフェース
  - 通信ライブラリの挙動や性能に関する解説と、性能向上支援ツール
  - 低オーバーヘッドでの実装

  これらの回答から、性能と利便性を両立させた通信ライブラリの提供が求められていることが改めて確認できた。

- DPU の試験環境整備

  強スケーリングの実現に向けて重要となる通信隠蔽技術について、近年、SmartNIC と呼ばれる、NIC に搭載した汎用プロセッサを用いる方法が提案されている。そこで今年度、この SmartNIC の一つである NVIDIA 社の DPU を購入して試験環境を整備した。また、それとは別に NVIDIA 社の DPU 搭載クラスタを PoC (Proof of concept)環境として試用できることとなった。今年度は、これらの環境を設定するとともに、基本操作を確認した。来年度以降、これらの環境で、DPU を用いた通信隠蔽の効果を検証する。

- 通信ソフトウェア課題表の作成

  本グループの成果をまとめるため、図 3.2.6 のような課題表を作成した。

課題		現有システムの状況			ポスト京に向けた具体的な要件	重要度	難易度と、他の技術への要求	関連グループ	ターゲットソフトウェア
		富岳、FX1000	ITO	システムB					
強スケーリング並列化支援	ノード内高速通信	?	MPIライブラリ（Intel, Open MPI, MVAPICH）依存	CMA	遅延時間 a 秒以内、バンド幅 b Gbit/s以上				
	ノード間高速通信	Tofu-D＋utofu	IB＋verbs	Slingshot＋GNI	遅延時間 c 秒以内、バンド幅 d Gbit/s以上				
	通信隠蔽	アシスタントコア	SHARP	-	ベンチマークプログラム e での隠蔽率 f %以上		Smart NIC、アシスタントコア、高機能スイッチ		
	アクセラレータ間直接通信	-	GPU Direct	-	遅延時間 g 秒以内、バンド幅 h Gbit/s以上				
ワークフロー実行	同一システム内連成計算	MPMD MPI	MPMD MPI	MPMD MPI	？？が可能な連成ライブラリ				
	システム間連成計算	WaitIO	フロントエンドノード群経由TCP,UDP	-	？？が可能な連成ライブラリ				
	外部高速通信	-	-	-	遅延時間 i 秒以内、バンド幅 j Gbit/s以上				
業界標準インタフェース	標準低レベル通信	Open Fabrics, UCX	Open Fabrics, UCX	UCX	Open Fabrics, UCX				
高セキュリティ	ノード間通信暗号化	-	無し	-	通信速度に対するオーバヘッド k %以内での暗号化		Smart NIC、		
	遠隔メモリ保護		IB protection domain	-					
高効率運用	耐障害性	-	MPIライブラリ依存（Intel, Open MPI, MVAPICH）	?	ULFMの実現		ULFM		
	チェックポイントリスタート	?	無し	?	？？が可能なチェックポイントリスタート機能				

図 3.2.6 通信ソフトウェア課題表

この表では、次世代計算基盤における通信ソフトウェアの課題を以下の 5 個のカテゴリに分類する。

- 強スケーリング並列化

  幅広い分野のアプリケーションの高効率な並列化のための高性能通信の実現

- ワークフロー実行

  演繹的シミュレーション、帰納的シミュレーション、データ収集処理、同化処理などの相互連携のための接続機能の提供

- 業界標準インターフェース

  国内外の様々なソフトウェアをそのまま利用可能な標準ライブラリ群の提供

- 高セキュリティ

  機密性の高いデータの保護技術の実装

- 高効率運用
  資源を有効に利用した高スループット運用

各カテゴリについて、達成に必要な課題項目を定義し、各項目の具体的な要件、重要度、難易度、関連するグループとターゲットソフトウェアを記述する。これにより、次世代計算基盤における通信ソフトウェアについて、開発項目の優先度決定や開発コスト見積もりを可能とする。

3.2.3.4 来年度計画

来年度は以下の調査を実施する予定である。

- 通信ソフトウェア課題表に基づいて、関連グループや組織へのインタビューを実施
  「次世代計算基盤に係る調査研究事業」のうち、特に通信ソフトウェアとの関連があるアーキテクチャ、運用、システム・ソフトウェア、アプリケーションの各グループを対象に、各課題の重要性や難易度を確認する。また、計算機センタのうち特に利用者向けチューニングサポートを手厚く実施しているセンタに対し、利用者プログラムにおける通信ソフトウェアの利用状況などを問い合わせる。さらに、低レベル通信レイヤおよび高レベル通信レイヤの開発グループに対し、国際ワークショップ等の機会を利用して、今後の方針などを問い合わせる。
- アプリケーションの調査
  アプリケーション中での通信ソフトウェアの利用状況を解析し、次世代計算基盤に求められる通信ソフトウェアの性能、機能、およびインターフェースについて調査する。来年度は、主に「次世代計算基盤にかかわる調査研究事業」のベンチマークグループが作成するプログラムを対象とし、非同期集団通信の現在の利用状況や、今後の需要の見通しなどを調査する。
- DPU による通信隠蔽効果の検証
  九州大学に導入した DPU 環境、および NVIDIA 社から提供された PoC 環境を用いて、DPU による通信隠蔽効果を検証する。

3.2.4 I/O・ストレージ・ファイルシステム

3.2.4.1 調査目的

次期ストレージアーキテクチャについてのフィジビリティスタディを行う。

3.2.4.2 調査内容

ストレージデバイスの動向調査、各種アプリケーションのアクセスパターンの調査、ストレージアーキテクチャの調査をもとに、次期ストレージアーキテクチャについての考察をまとめる。

3.2.4.3 調査結果

このレポートは2030年頃までのストレージシステムの動向をまとめたものである。まず、ストレージデバイス、インターフェース、ネットワークの技術動向、ストレージアクセスパターン、ストレージアーキテクチャの動向についてまとめる。これらの調査を元に、次期ストレージシステムについての考察をまとめる。

3.2.4.3.1 デバイス・インタフェース・ネットワークの技術動向

3.2.4.3.1.1 HDD

2022年においてデータセンタ用のHDDでは18TBがボリュームゾーンとなっており、22TBまでの容量の

ものも市場に出ている。消費電力は9W程度である。今後、MAMR (Microwave Assisted Magnetic Recording)、HAMR (Heat Assisted Magnetic Recording) により容量が増加していく。2026年頃には50TB超のHDDの出荷も予想される。また複数のアクチュエータをもち、IOPSの向上を目指したものも考えられている。容量当たりのコストはSSDに比べHDDが一桁ほど優位となり続ける。

3.2.4.3.1.2 SSD

2022年においてエンタープライズ用のSSDでは30TBまでの容量のものが市場に出ている。PCIe Gen5で接続するNVMe SSDについても試作品が出荷されている。また遅延が小さいフラッシュストレージの開発も進められており、2022年においては、読込みは29マイクロ秒、書込みは8マイクロ秒の製品が出荷されている。SSDはフォームファクタがEDSFF (Enterprise and Datacenter Standard Form)へと変わり、優れた放熱設計を可能とする。

2022年現在はセル当り3ビットを保持するTLCが主流であるが、4ビット保持するQLCも増えている。QLCになると容量当たりのコストが10%ほど下がる一方で、ランダムアクセス性能が低下する問題もある。2024年頃には5ビット保持するPLCの試作機が登場し、本格的の出荷は2026年頃が予想される。また、SSDはフラッシュの積層数を増やして容量が増えていく。2024年頃には120 TBのSSDの出荷も予想される。

PCIeの世代に応じてそのインターフェースに対応したSSDも出荷され、2025年頃にPCIe Gen6、2028～2030年頃にPCIe Gen7対応の製品出荷が始まると予想される。PCIe Gen6、Genインターフェースのバンド幅は倍増し、それぞれ16 GB/s、32 GB/sとなる。一方で、PCIeの世代が上がるにつれ性能を向上させるためコントローラの消費電力が増えていく。PCIe Gen4では20Wほど、Gen5では25～30Wほどである。

3.2.4.3.1.3 PCIe

PCIe Gen6の標準化は2022年に行われ、PCIe Gen7の標準化は2025年頃が想定されている。対応する製品の出荷はその3～5年後くらいが予想される。

3.2.4.3.1.4 CXL

本章では、データインテンシブコンピューティング領域において、コンピュータが取り扱うデータを配置する方法の一つとして、CXL等により接続されたDRAMやフラッシュメモリを活用することを検討する。従来、コンピュータがデータを取り扱うためには、I/Oによりストレージデバイスからメモリにデータを読み込む必要があったが、CXL等の技術によりメモリを柔軟に運用できるようになったことから、データ共有の仕組みとしても使用できる可能性が生じており、ストレージ領域においても本技術の使用を検討すべき状況にある。

メモリプーリングと共有メモリ

CPUとメモリの間の接続は、近年の計算機では1台の計算機内で完結し、固定した構成が取られているが、CXLをはじめとする技術により動的な再構成やデータ共有が可能になる見込みである。本節では、この技術により提供される機能のうち、メモリプーリングと共有メモリについて説明する。

メモリプーリングは、別のノードあるいは専用の筐体に搭載されたメモリを、必要に応じて自ノードに接続する仕組みである。接続されたメモリは、CPUを持たないNUMAノードの1つとしてOSから認識され

る [1]。この形態は、あるノード上でノードが持つメモリ以上の容量を必要とするアプリケーションを動作させる際に使用することができる。このため、少数の大メモリを必要とするアプリケーションのために大量のメモリを各ノードに搭載する必要がなくなるため、メモリの総量を削減し、消費電力やコストの削減に貢献できる可能性がある。あるHPCシステムにおいては、運用時間のうち8割の時間において、75%のメモリが空きであった事例も報告されており[2]、メモリ総容量の削減余地は無視できない大きさであるといえる。このような状況の一例としては、並列プログラムにおいてランク0にのみ前処理や集約処理が存在し、それらが多量のメモリを消費する一方で、他のランクではそれほどのメモリを消費しない場合が挙げられる。メモリプーリングを用いてリソースの割り当てを最適化するための既存技術としては文献[3]がある。これはクラウド環境において、VMのメモリ割り当ての一部をメモリプールから行うことにより、性能影響を抑えながらも合計メモリ量を減らす手法について述べている。また、文献[4]では、HPCアプリケーションをメモリプーリング環境で動作させた場合の性能特性について述べており、リモートメモリを用いても必ずしも大幅な性能低下が起こらないことが示されている。

共有メモリは、複数のノードから同一の領域を読み書き可能にする形態である。特定のメモリ領域を共有領域として設定すると、複数のノードから同時に読み書きが可能になる。この形態は、複数のアプリケーション間におけるデータ共有に用いることが可能で、従来の通信を用いた遠隔メモリアクセスの実装と比較すると、大幅に短い時間で読み書きを完了することができる。

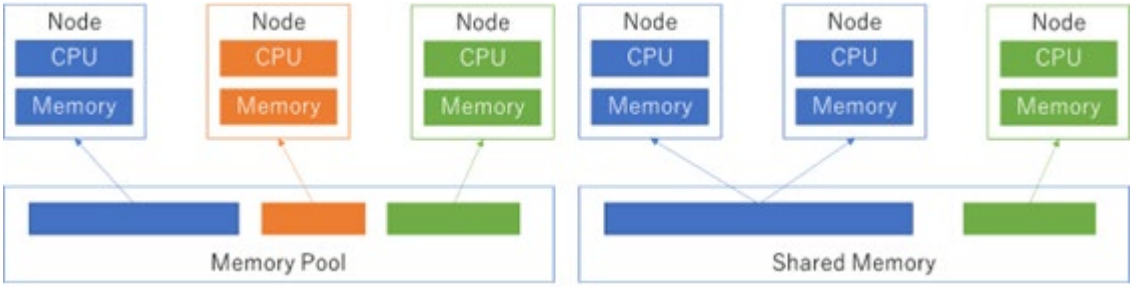


図 3.2.7 メモリプーリングと共有メモリの接続概念図

CXLによるメモリ接続

本節では、前節に記したメモリプーリングと共有メモリを実現するための技術について説明する。CXLはPCI Express上においてメモリなどを取り扱うための規格であり、現在バージョン3.0が公開されている。CXL1.1ではアクセラレータやNICを対象としていたが、CXL2.0以降ではメモリデバイスもサポートしている。CXL2.0ではメモリプーリング、CXL3.0 [5]では共有メモリもサポートされる。CXL2.0規格は2020年12月にリリースされ、2022年には各ベンダから製品デモが行われている状況であり、2023年中の製品登場が期待される状況である。尚、CXL3.0規格は2022年8月に公開されている。

メモリプーリングにおいてリモートのメモリにアクセスする際のレイテンシオーバーヘッドは、CXL由来の分はNUMAノード1ホップ相当[6]、筐体をまたぐ場合などはスイッチの処理時間や配線長による遅延が上乗せされる。具体的なレイテンシは[7]によると、通常のメインメモリが80～140ナノ秒に対してCXLメモリは170～250ナノ秒とされている。メインメモリよりは多少レイテンシは増大するものの、SSDの10～40マイクロ秒に対しては2桁近く短縮されることになる。

共有メモリは複数ノードからアクセスされるため、コヒーレンシコントロール[5]も必要である。これについては、CXL3.0ではハードウェア制御とソフトウェア制御を選択することが可能である。まだこのハードウェアは存在しないため仕様は明らかではないが、ハードウェアは高速な制御が可能である一方仕様

上の同時接続最大数などが存在する一方、ソフトウェア制御では性能はハードに及ばないにせよ接続数などにおいて柔軟性が高いと考えられる。これらは今後の実装の進展により明らかとなっていく部分である。

活用ケースと効果

高性能計算領域におけるワークロードは、従来の計算が主であるものと、データの取扱量が多いいわゆるデータインテンシブ系に分類される。本節では主に後者について、共有メモリ領域の活用ケースを検討する。

近年、自然言語学習モデルの巨大化や、大規模グラフデータの活用が進んでいる。通常の分散メモリ型のコンピュータで特にこれらのモデルやデータを利用する場合、同じデータを多数のノードのメモリ上に配置する状況や、異なるアプリケーションを動作させるたびにストレージからメモリにデータをロードする状況が発生する。このような状況において共有メモリが存在すると、データ全域に対して常にアクセスがある訳ではないが、データ量が大きくかつ低レイテンシ性が求められるような処理を効率的に行える可能性がある。例えばグラフデータの探索を共有メモリ上で行うことにより、全ノードへの同一データのコピーを回避することが考えられる。また、推論用モデルは通常ストレージから読みだして、アクセラレータのメモリに転送されるが、これらのデータを共有メモリ上に配置することで、CXLメモリ・アクセラレータメモリ間のコピーに置き換えることで高速化し、アクセス高速化を目的としたローカルストレージ等へのデータ複製を抑制できることも期待される。

以上のように、CXLなどにより接続されたメモリ領域は、プーリングによりメモリ領域の拡張を行うことや、共有メモリにより低レイテンシアクセスが可能なデータ領域の提供に用いることができ、コンピューティングにおけるデータの配置方法を変えることで、さらなる効率化と高性能化を期待できる技術であるといえる。従来のストレージと協調して活用することにより、今後増加が見込まれるデータインテンシブアプリケーションの要求にこたえる上で有用であるといえる。

[1] Maruf, H., et.al., TPP: Transparent Page Placement for CXL-Enabled Tiered Memory, Arxiv, abs/2206.02878, 2022
[2] Peng, I., et.al, A Holistic View of Memory Utilization on HPC Systems: Current and Future Trends, MEMSYS 2021, 2022
[3] Li, H. et.al., Pond: CXL-Based Memory Pooling Systems for Cloud Platform, ASPLOS 2023, 2023
[4] Wahlgren, J., et.al., CXL-enabled Memory Pooling for HPC Systems, 2022 IEEE/ACM Workshop on Memory Centric High Performance Computing (MCHPC), pp. 11–20), 2022
[5] CXL Specification, https://www.computeexpresslink.org/download-the-specification (2023年3月参照)
[6] Ahn, M., et.al., Enabling CXL Memory Expansion for In-Memory Database Management Systems, DaMoN'22, 2022
[7] Chauhan, P., et.al, CXL Memory Challenges, Hot Chips 34, 2022

3.2.4.3.1.5 InfiniBand

400 GbpsのInfiniBand NDRは2022年に出荷された。800 Gbps のXDRについては2023年以降に規格が制定され2024年以降に登場すると思われる。2028年～2030年であればその次の1.6 Tbps の規格が利用可能になると考えられる。

3.2.4.3.2 ストレージアクセスパターンの調査

3.2.4.3.2.1 科学技術計算アプリケーション

科学技術計算アプリケーションは、入力データを読込み、計算結果を定期的に書き出すパターンが多い。読込み、書込みではプロセス毎にファイルを読み書きするFPP (File Per Process) と、単一ファイルを読み書きするSSF (Single Shared File) がある。SSFについてはHDF5、NetCDF等の高水準並列I/Oライブラリを用いることが多い。高水準並列I/Oライブラリでは、プロセス間で分散して保持している複数の配列を単一ファイルに保持することができる。アクセスパターンは配列ごとにブロックストライプとなるが、ブロック長は配列のサイズとプロセスのトポロジによる。FPPはプロセスの数、配列の数だけファイルが生成されるため、大規模アプリケーションではSSFが主に利用される。FPPの場合は、アクセスする複数のファイルが異なるストレージに格納されていることが多く、ストレージのバンド幅を容易に活用することができる。一方で、SSFの場合は、同一のストレージに異なるプロセスがアクセスすることがあり、ロックの競合等によりバンド幅を活用することは一般に難しい。ストレージシステムで解決する必要がある。

科学技術計算アプリケーションの場合、並列アプリケーションの全プロセスが一斉にファイルオープンを行う。FPPの場合は別々のファイルのオープンとなり、SSFでは単一のファイルオープンである。そのため、プロセス数に応じたファイルシステムのメタデータ性能が要求される。

富岳のユーザについては、POSIXによるFPPパターン、NetCDF、PnetCDF、MPI-IO、HDF5によるSSFパターンの両方が存在する。現行の富岳のストレージシステムについては、ディレクトリ当りのファイル数が多いとエラーとなる問題、SSFアクセスの性能が低い問題、実行プロセス数が多くなると性能低下、タイムアウト等によるエラーが発生する等の問題があり、次期ストレージシステムではこれらの改善が必須である。

3.2.4.3.2.2 深層学習

深層学習ではサイズの小さな多くのファイルに対しランダムにアクセスする。しかしながら、ストレージアクセス性能を向上させるため、複数のファイルをまとめてファイルサイズを大きくする等の最適化が用いられる傾向にある。また、深層学習のモデルは大きくなる傾向にあり、その場合計算コストに対するファイル入出力コストの比率が下がり、ストレージシステムに対する性能要求は下がる傾向にある。

一方、深層学習を行うためのデータの整備については全データについて探索的データ解析、キュレーション等の前処理が必要となる。これらの処理については計算に対するファイル入出力の比率が大きいため、高いストレージ性能が要求される。

3.2.4.3.3 ストレージアーキテクチャの調査

3.2.4.3.3.1 Lustre

本節では、「富岳」でも利用しているFEFSのベースファイルシステムであるLustreについて調査した内容について報告する。

Lustreはオープンソースソフトウェアの並列ファイルシステムでGPLv2にて公開されている。その歴史は古く20年以上に渡り開発が継続されており、最新リリースはlustre-2.15.1(2023年3月時点)で継続

的に新機能が追加されている。

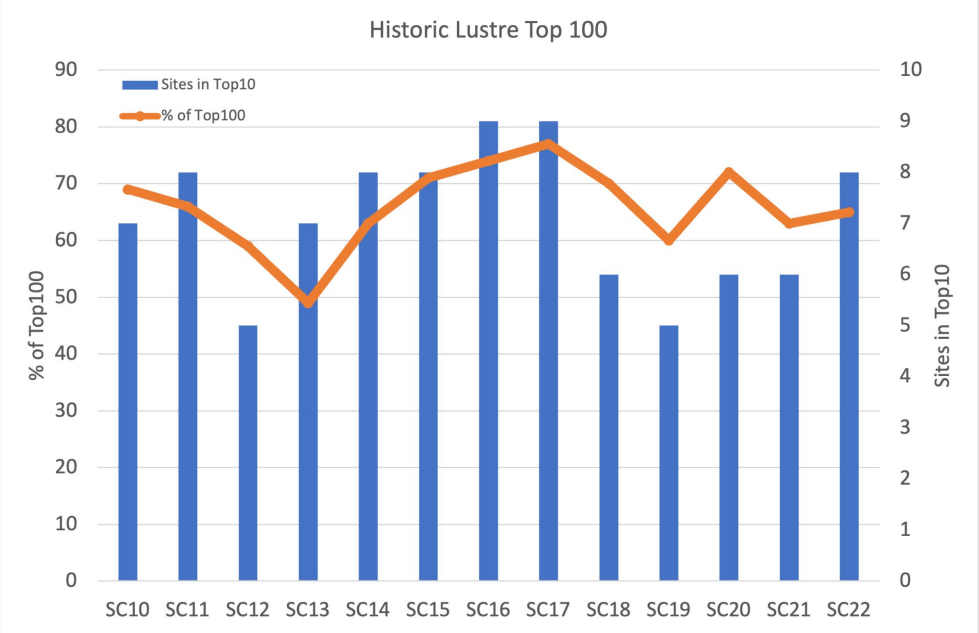


図 3.2.8 過去 13 年間の Lustre の採用したシステムの割合

図 3.2.8にTop500におけるTop10とTop100にてLustreを採用したシステム数および割合を示す。最新のリスト(SC22)においてTop10の8システム、Top100においても70%を超えるシステムがLustreを採用している。また、大規模なHPCシステムの多くでLustreが利用されていることが分かる。

採用の理由として挙げられるのが

- 数万ノードを越えるクライアント数、シングルネームスペースとして 100PB、1 千億を超える inode 数での運用実績
- 様々な CPU アーキテクチャ(x86_64、Power、ARM)および GPU をサポート
- Ethernet (RoCE)、InfiniBand をはじめベンダ固有のネットワーク(Tofu、slingshot)もサポートし、それらを複合的に接続する LNET ルーターも提供
- ファイルシステムの合算性能に制限はなく、OSS/OST 追加によりリニアに性能向上

次節以降、現在最新バージョンのLustreが提供している機能で大規模HPCストレージに関連しているものを中心に説明し、将来の課題およびその課題に対する取り組みを述べる。

最新のLustreアーキテクチャ

Lustreはファイルのメタデータを格納し管理するMDS/MDTとファイルのコンテンツを格納し管理するOSS/OSTに分かれているが、昨今では性能、耐久性を含めタイプの異なるストレージメディアが存在するため、それらを意識した構成が必要になる。

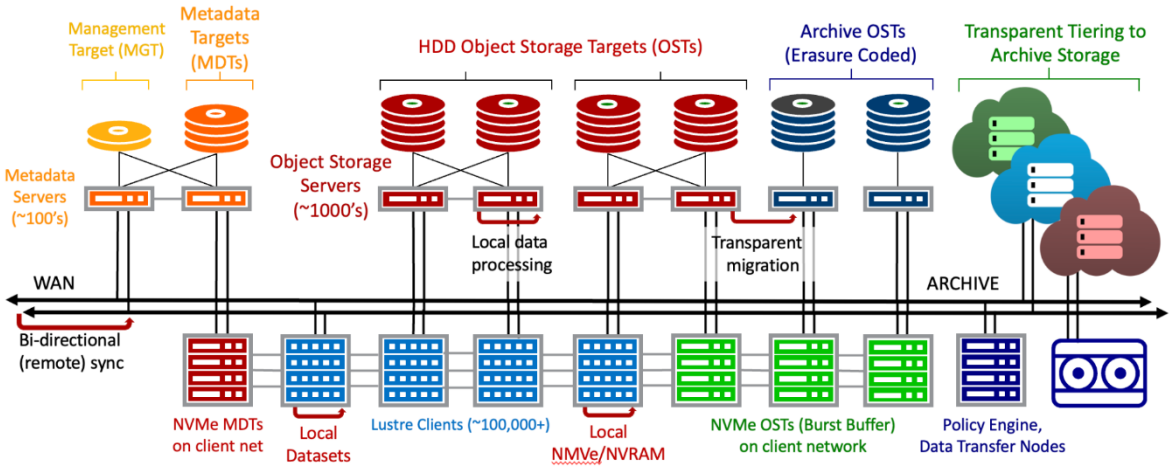


図 3.2.9 ハイブリッドストレージと Lustre

図 3.2.9に示すのが、複数のストレージメディアで構成されたOSS/OST環境を単一の名前空間として構成した例である。下記のそれぞれの機能について説明する。

a) Auto Tiering Between NVMe and HDD OSTs

NVMeが持つ性能とHDDを利用した安価な大容量ストレージの両方を実現するには、ハイブリッドストレージ構成が不可欠で、アプリケーションに対して、透過的なデータアクセスを実現する HDDとNVMe OST間のデータ管理が重要になる。DDNのHot PoolsではLustreのFLR(File Level Redundancy)を利用した透過的な機能を実現している。Hot Pools(2021)ではクライアントからNVMe OSTに書き込まれたファイルに対し、複製ファイルをHDD OSTに非同期で作成する。この際、メタデータにはHDD OSTに作成された複製OSTオブジェクトが登録される。ファイルへのアクセス頻度を定期的に確認し、ファイルアクセスが少ない場合、NVMe OST上の複製を削除し、NVMe OSTの容量を再確保する。ファイルのメタデータは、自動的に更新され、次回以降のファイルアクセスはHDD OSTに対して行われる。これにより、直近にアクセスされたユーザのファイルはNVMeからアクセスし、長期に渡りアクセスがないファイルはHDD OSTへ自動的に退避することが可能になる。

b) Persistent Client Cache (PCC)

PCCは、計算機に搭載されている永続的ディスクを利用したキャッシュ機能である。計算処理中、一度アクセスしたファイルを永続的ディスクにPCCとして保存し、次回以降アクセスする場合、PCCを利用することで不要なネットワーク資源およびレイテンシ、OSTアクセスを排除することができる。PCCは計算機におけるページキャッシュと同様、ファイルに対して透過的にキャッシュされるためユーザは意識することなく利用可能である。PCCは、主にReadキャッシュとして利用されるがWriteキャッシュもサポートしている。ただし、複数の計算機にてデータを共有する場合は、PCCのダーティデータをすべてOSTに書き込み後、他のクライアントがアクセスすることとなる。この間、PCC対象のファイルへのアクセスはブロックされるがファイルへの一貫性と透過性は保証される。

c) アーカイブストレージ

LustreはHSM (Hierarchical Storage Management) 機能を有しており、単一の名前空間を維持したままデータを低速ディスク、テープ、クラウドストレージなど外部ストレージにアーカイブすることが可能である。Amazon FSx for Lustreは、AWSのLustreサービスにてS3オブジェ

クトストレージをアーカイブ領域として利用し、HSMによって透過的なデータ管理を実現している。低コストで大容量を実現するためには、アーカイブストレージは重要になる。また、LustreのHSMは拡張性に優れており、それぞれの外部ストレージに対応したコピーツールを開発すればアーカイブストレージとして利用可能である。HPCIで利用されているGfarmファイルシステムへのアーカイブも可能で、次期富岳においては、外部ストレージへの連携も含めたアーカイブストレージを検討する必要がある。

その他の機能

ここでは現在開発中、または今後開発予定のLustre機能にて説明する。

(注: ここに示すLustreバージョンと時期は予定なく変更される可能性がある)

a) メタデータサーバのスケーラビリティ

Lustre DNE (Distributed Namespace Environment) によって、1 つのファイルシステム内で、複数のMDS/MDTを利用できる。これまで、管理者が任意にMDTを割り当てる必要があったが、lustre-2.15/lustre-2.16で実装されたDNE3のAuto space balancing機能によって複数のMDTを自動的にロードバランシグすることが可能になり、MDS/MDTを追加することによって、ファイルシステムのメタデータ性能がリニアにスケールする。現在は48台までのMDT数の制限はあるが、128台のMDTまでは将来容易に拡張可能である。1台のMDSあたり100K/秒のファイル作成能力を有するメタデータサーバを利用した場合、最大10M/秒のファイル作成能力が可能になる。

b) メタデータのローカリティ

Lustreには任意のOSTをグループ化し、任意のディレクトリに割り当てるOST Pool機能がある。これを活用することで、クライアントからストレージまでのI/Oパスを最適化でき、OSTアクセスのローカリティを可能にする。現在MDT poolは実装されていないが、本機能の実装により、OST同様、複数のMDTをグループ化したpoolをディレクトリに割り当てることで単一の名前空間の中でMDS/MDTの分離および局所化が実現できる。

c) Batched Metadata RPCとWriteBack Cache

Lustreにおける単一ディレクトリに対する複数のメタデータオペレーションは同時に可能だが、それぞれ異なるRPCとしてクライアントからサーバに発行される。大量のメタデータ操作があった場合に、メタデータ操作のコストはN x RPC往復レイテンシー(ネットワークも含む)に依存する。Lustre-2.16(2023)ではBatched Metadata RPCを実装し、複数のメタデータ操作を集約し、単一のRPCとしてサーバに発行できる機能を実現する。これにより、クライアントにて単一プロセスから発行される、連続したメタデータ操作の高速化が可能になる。しかし、単一プロセスではCPUハードウェアスペックに律速するため、今後は並列化されたツールを活用していくことも合わせて検討する必要がある。Batched Metadata RPCで実装したフレームワークを活用し、Lustre-2.17(2024)/Lustre-2.18(2025)以降で実装が予定されているのが図 3.2.10に示すMetadata WriteBack Cacheである。

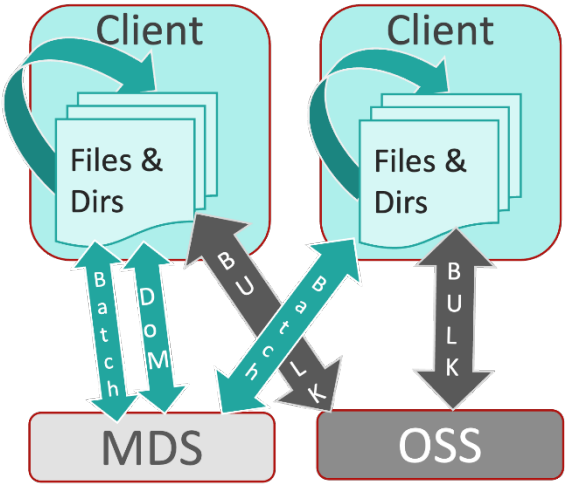


図 3.2.10 Metadata WriteBack Cache

Metadata Writeback Cacheでは、一定量または一定期間のメタデータ操作をクライアントのメモリ内で実行し、MDSとのやりとりは即座には行わず非同期でバッチ処理される。ローカルクライアントで処理できるメタデータ操作や即座に他クライアントとの共有が必要ないメタデータ操作に対してWriteBack Cacheを適応していくことで、限られたサーバおよびネットワーク資源の有効利用ができる。また、ローカルで処理されるメタデータ操作は、いずれもクライアント上のメモリで処理されるため超高速なメタデータアクセスが可能となる。

d) メタデータ冗長化

単一名前空間にて、多数のMDS/MDTをDNEで構成した場合のメタデータの可用性および冗長化の重要度が上がる。LustreはMDS/MDTのHA (High Availability)をサポートしているため、サーバまたはストレージに障害があった場合に、自動的に縮退運転となりファイルシステムサービスの継続が可能である。

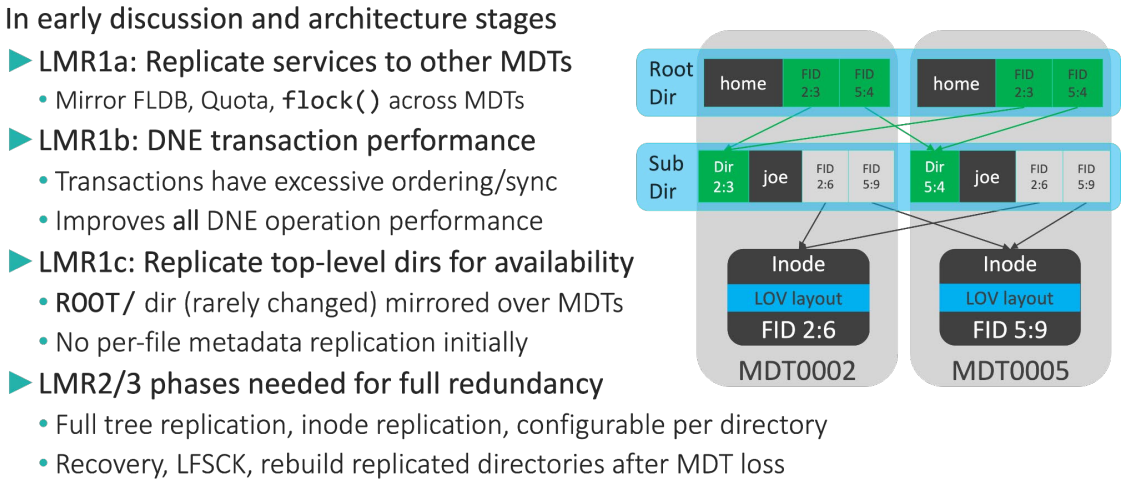


図 3.2.11 Lustre メタデータの冗長化

しかし、LustreのHAはLustreメタデータ自体の複製ではないため、Lustre-2.18(2025)以

降、メタデータの冗長性を実装する予定である。

図 3.2.4図 3.2.11に示すとおり、メタデータの冗長化の実装は複数のフェーズに分かれている。

第１フェーズでは現在MDTで機能しているサービスの複製を他のMDTでも可能とし、DNEにおける全てのトランザクション性能の向上を目的とする。その上で、トップディレクトリの冗長化を実装し、最終的にはサブディレクトリを含むすべてのディレクトリおよびファイルメタデータの複製を実現する。

e) Luster CCI (クライアントコンテナイメージ)

AIや学習のプラットフォームにおいては、ライブラリやバイナリを含む実行環境をコンテナイメージで管理することが主流である。コンテナで管理することにより、更新差分をスナップショットしソフトウェア間の互換性問題を解決したり、実行環境の堅牢性、ポータビリティが大きく向上する。ユーザ空間で実行できるコンテナ(Singularity等)はMPIもサポートしており、今後HPCにおいても実行環境のコンテナ化の普及が予想される。また、実行環境のみならず、データセットも共通のフォーマットのコンテナで管理することでデータに効率良くアクセスが可能になる。Lustre-2.19(2026)以降、統合化されたLustreコンテナイメージのサポートが計画されている。現在でもLustreクライアント上にイメージファイルを作成しloopbackとしてマウントしイメージファイル内にファイル作成および参照が可能であるがユーザが透過的に利用することは複雑である。

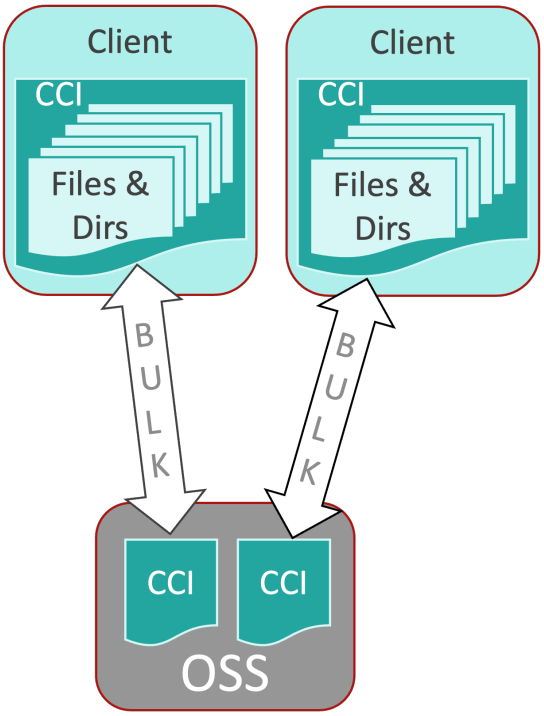


図 3.2.12 Lustre CCI

Lustre クライアントコンテナイメージは、イメージ管理のインターフェースも含めLustreに統合することを目的とする。本機能により、ユーザが任意に設定するLustreストライプの設定同様、ディレクトリにコンテナイメージを利用するように指定することでLustreがコンテナイメージの作成およびクライアント上のマウント等の管理を行いユーザは透過的なアクセスが可能になる。コンテナイメージは、複数のクライアントからの排他的なアクセス、複数のクライアントからのRead onlyアクセス、または将来的にMDTの内部的なディレクトリにイメージを利用することで複数クライアントからのRead Writeアクセス等が予定されている。また、コンテナイメージは、OST上に配置されるためコンテナ内

のメタデータアクセスをOSTのBulk IOとして処理することができるので、メタデータ性能の向上に役立つと同時にメタデータ資源の削減に大きく貢献でき、データのバックアップやアーカイブ等にも非常に有用なデータ管理手法になる。

3.2.4.3.3.2 DAOS

インテルは、富岳NEXTプロジェクトにおける次世代スーパーコンピュータの高性能ストレージソリューションとしてDAOSを提案する。DAOSはオープンソースプロジェクトであり、インテルのコアエンジニアリングチームだけでなく、アカデミックパートナーと産業界パートナーの両方によるx86_64およびARM64プラットフォームコミュニティへの貢献も大きい。このオープン性は、プロプライエタリなストレージソリューションと比較して戦略的優位性を持っている。国際協力を可能とし、DAOSを元にした差別化されたストレージソリューションの開発にもつながる可能性がある。大規模なHPC/AI顧客や特定の科学分野のためにDAOSの機能を共同設計することは、当初からDAOSエンジニアリングチームの基本方針だった。例えば、今日のDAOSソフトウェアの機能の多くは、Auroraエクサスケールシステム調達の一環として、アルゴンヌ国立研究所（ANL）と共同設計されたものである。富岳NEXTのフィジビリティスタディでは、選定されたスーパーコンピュータとストレージソリューションを最適に統合するために、新しい機能の開発が必要な領域がさらに特定されることが期待される。そのため、今後2～3年のDAOS開発ロードマップはDAOSコミュニティWiki（https://wiki.daos.io/spaces/DC/pages/4836661105/Roadmap） で公開されているが2028～2030年のタイムフレームにおけるソフトウェア開発の優先順位は、DAOSを極めて大規模に展開することを意図した富岳NEXTのような大規模プロジェクトの要件に大きく影響されることになる。

DAOSは主にソフトウェアプロジェクトであり、ハードウェア基盤にとらわれないが、システムアーキテクチャサブグループで検討されているハードウェアソリューションを意識する必要があり、想定される2028-2030年の展開スケジュールに関連する：

- ノードレベルのファブリック統合とストレージデバイスの性能： 専用ストレージサーバには、DAOS は汎用的な 1U または 2U のラックサーバを使用する。4 x 400Gbps ファブリックポートと十分な数の PCIe gen5 NVMe ディスクがあれば、サーバあたり最大 180GByte/s のストレージ帯域幅を提供する。産業界では、U.2 NVMe ディスクから E1.S（1U サーバ用）または E3.S（2U サーバ用）への移行が進んでいるが、今後数年間はハードウェアの基本性能特性に大きな変化はないと予測される。富岳 NEXT の展開時期には、PAM4 シグナリングの PCIe gen6 が利用可能になると思われる。これは HPC ファブリック側で活用され、DAOS サーバあたりの外部帯域は 2 倍になるだろう。また、DAOS エンジニアリングチームは、いくつかの OEM ストレージベンダと協力し、DAOS ソリューションにおける実現可能性について、新しい CXL ベースのストレージデバイスを評価している。これには、従来の NAND フラッシュベースのデバイス（TLC、QLC、PLC）に加え、CXL.io と CXL.mem の機能を 1 つのデバイスに統合したより機能豊富なデバイスの両方が含まれている。

  システムレベルのファブリック統合とトポロジへの対応： 現在、DAOS は、Dragonfly トポロジの HPE Slingshot を搭載した 2 ExaFlops の Aurora マシンに導入されているほか、もっと小規模な InfiniBand、Ethernet、OmniPath システム上にも導入されている。2028-2030 年のタイムフレームでは階層化されたネットワークアーキテクチャが想定され、ポッドレベルまたはラックレベルの非常に高性能なインターコネクトとシステムレベルの低性能なインターコネクトを組み合わせ、システムの総コストを最小化する。既存の DAOS Exascale Architecture を拡張し、2 層モデルを導入することで、このようなより階層化されたネットワークトポロジに対応することが計画されている。グローバルにアクセス可能な単一の名前空間内に、ラックローカルの DAOS キャッシュ層（データ

転送のためにポッドレベルまたはラックレベルのファブリックに接続されたサーバ）と、外部の DAOS ストレージ層（システムレベルのファブリック上でラックレベルの層を介して接続）が存在することになる。DAOS はすでに（部分的に）トポロジへ対応しているが、このような 2 層ストレージアーキテクチャを最大限に活用するために必要なステージイン/ステージアウト機構を実現するためにはさらなる開発が必要である。

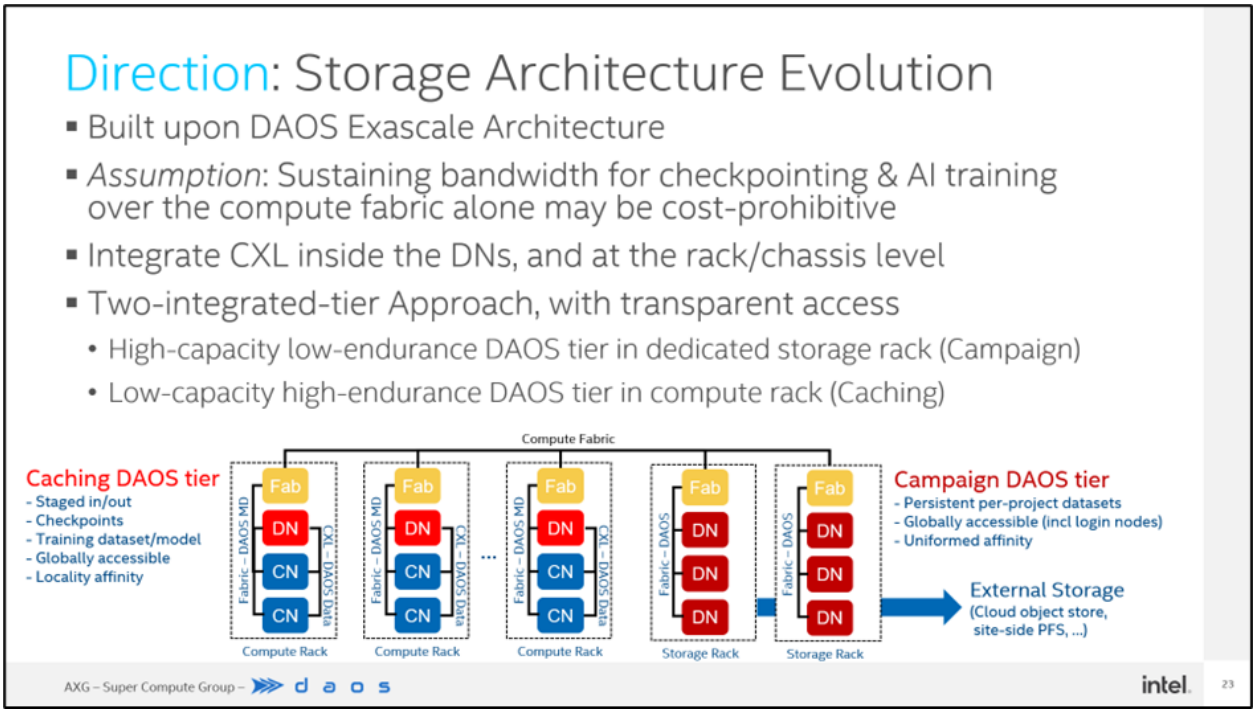


図 3.2.13

システムソフトウェアレベルで見ると DAOS ストレージアーキテクチャは従来の HPC ストレージシステムとは大きく異なっている。その主要な差別化機能はエクサスケール以降のスーパーコンピュータに理想的なものである：

- DAOS の一貫性モデルは、ログ構造のマージツリーの代わりに、独自の Versioning Object Store（VOS）を使用する。これは、POSIX だけでなく、ネイティブの DAOS Object API にマッピングできる他のドメイン固有のデータモデルの両方に対して、大きなパフォーマンス上の利点を提供する。DAOS ストレージアーキテクチャの機能を十分に活用するためには、ユーザはすでに DAOS バックエンドを持つ既存のフレームワーク（MPI-IO、HDF5、Tensorflow-I/O、PyDAOS など）を使用することが推奨される。世界をリードする主要なスーパーコンピュータプロジェクトではアプリケーション固有の DAOS コンテナタイプを通じてドメイン固有のデータセット管理の DAOS コンセプトを採用することも有益であり、その結果として実行性能が桁違いに改善されることが期待できる。
- DAOS は完全にユーザ空間で動作し、従来の OS のストレージスタックをバイパスして最新の NVMe および SCM ストレージデバイスのハードウェア性能を最大限に引き出すことができる。DAOS File System（DFS）API を直接使用できない従来の POSIX ベースのアプリケーションのためには、POSIX リード/ライトコールをインターセプトする I/O インターセプトライブラリが存在する（ただしメタデータ管理のために dfuse-mount に依存する）。このライブラリはまもなく改良さ

れた実装に置き換えられ、ユーザ空間の POSIX を完全にサポートし、メタデータ操作を含むすべての POSIX コールをインターセプトし、userfaultfd を介して mmap のサポートも提供する。

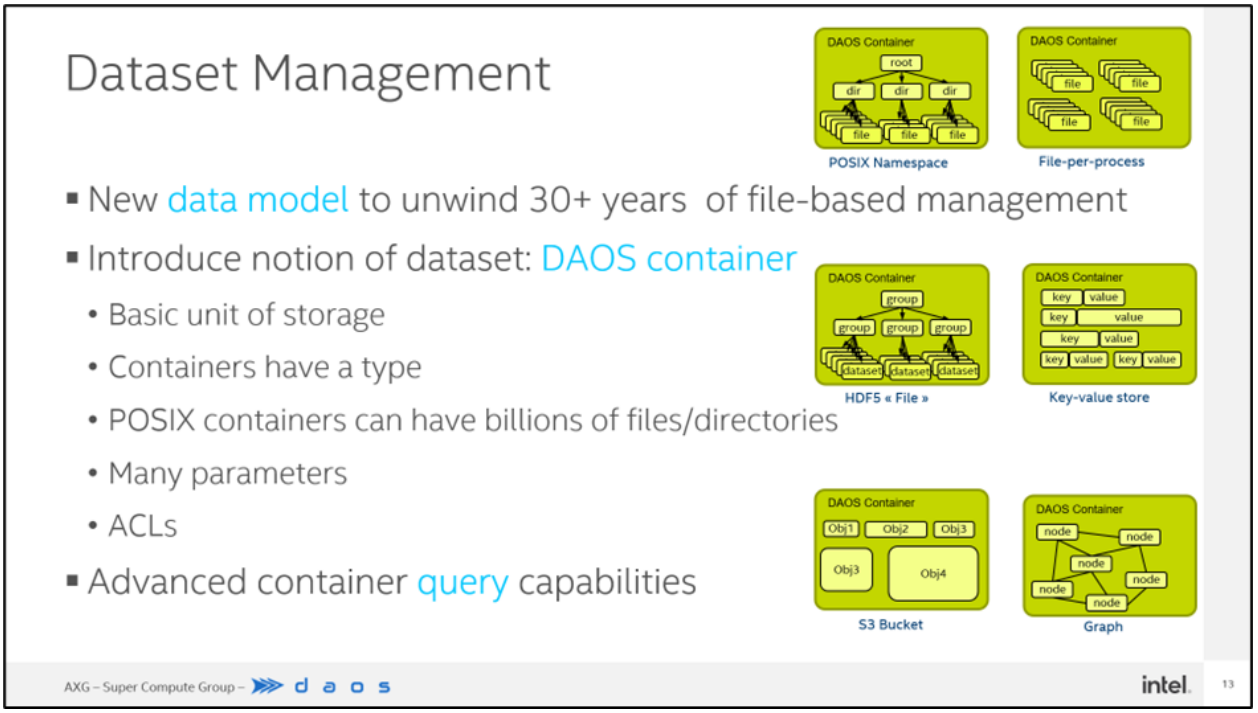


図 3.2.14

上記の（既に広く使われている製品機能または開発中である）ソフトウェア機能に加え、高性能ストレージスタックに、ストレージサーバからクライアントに転送するデータ量を減らすためにサーバ側でデータのフィルタリングや前処理を行う技術であるコンピュテーショナルストレージ機能を導入することが極めて重要であると考える。DAOS エンジニアリングチームはすでに、サーバ側の名前空間トラバーサル（POSIX find）のプロトタイプ実装を示し[1]、大規模なスケールアウトストレージシステムにおけるこのアプローチの実現可能性を実証している。このプロトタイプに使用された DAOS Pipeline API は今日の DAOS 製品のコードベースで利用可能であるが、一般目的のサーバサイド処理には簡単に使用できない。

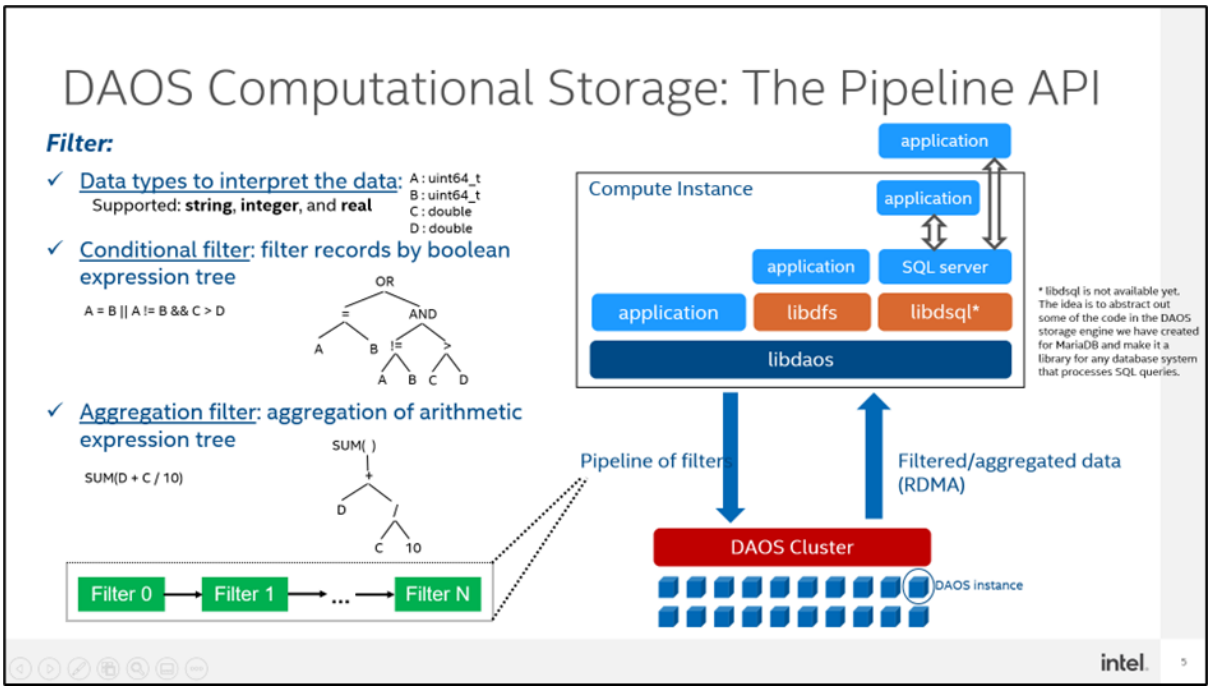


図 3.2.15

DAOS エンジニアリングチームの目標は、このコンセプトを一般化することある。 ユーザはフィルタリングと前処理ルールを高いレベルで定義することができ、それがバイトコードとして DAOS サーバに送られ、サーバサイドで処理されるようにする必要がある。このようなフレームワークは、ユーザ定義のフィルタリングのためのロバストなコンパイルとランタイム環境を確立した eBPF（enhanced Berkeley Packet Filter）インフラストラクチャをモデル化することができる。この技術はもともとネットワークパッケージの検査/フィルタリングのために開発されたが、その後、パフォーマンス監視を含むより多くのユースケースを持つ、より多用途なツールに成長した。

最初の「find」ユースケースは、ファイルの作成時間やアクセス時間、ファイルサイズなどのオプションの述語を使って、大きな POSIX 名前空間を検索することである。Pipeline API を使わない現在の実装では、クライアント上の「find」アプリケーションは、readdir()を使用してすべての名前空間エントリを読み込む必要があり、次にエントリのタイプ（ファイルまたはディレクトリ）、ctime/mtime、ファイルサイズなどを決定するためにすべてのエントリに対して stat()を実行する。Pipeline API を使用する場合、クライアントアプリケーションは、ファイル名の正規表現と ctime などの他の条件を含む述語を作成する。そして、DAOS サーバに「readdir() with predicate」リクエストを送り、DAOS サーバは述語にマッチした結果のみを返す。クライアントアプリケーションは、ファイルサイズを決定するために、より少ない数のマッチした結果に対して stat()を実行することのみ必要である。

このような計算ストレージ基盤のもう一つのユースケースは、データの前処理である。多くの AI アプリケーションでは、データ型の変換、データの正規化、テキストの整形・処理など、ストレージサーバにオフロードすることが可能な前処理を行う必要がある。今日のストレージシステムでは、データセット全体をクライアントノードに読み込み、データを変更した後、データを不揮発メモリに書き戻す必要がある。DAOS Pipeline API を拡張することで、データセット全体をクライアントに転送して書き戻すことなく、サーバ側でこれらの変換を実行することができる。

参考文献：

[1] Chaarawi, M, “DAOS Features for Next Generation Platforms”, ISC 2022 IXPUG Workshop, 2022, https://www.ixpug.org/events/isc22-ixpug-workshop.

3.2.4.3.3.3 LLIO

本節では、「富岳」向けに富士通が開発したファイルシステムであるLLIOの開発背景や機能について説明したのち、「富岳」運用における課題とその改善案について報告し、最後に富岳NEXTに向けての検討事項を述べる。

「富岳」のストレージ概要とLLIOの開発背景

LLIOは、「富岳」向けに富士通が開発したジョブ実行専用の高速ファイルシステムである[1]。「富岳」のストレージは三階層構成になっており、それぞれのストレージのファイルシステム、およびその用途は以下の通りである。

- 第一階層ストレージ：LLIO
  「富岳」で実行されるジョブのファイル I/O を高速に処理するファイルシステムである。「富岳」で実行されるジョブのファイル I/O は、基本的[1]に 全て LLIO で処理される。第一階層ストレージ上のデータはジョブ実行中のみ保持されるため、保存ファイルの入出力を行う場合は、後述する第二階層ストレージのキャッシュ領域を利用する。
- 第二階層ストレージ：FEFS[2]
  「富岳」を利用するユーザおよびジョブがアクセスする大容量（約 150PB）の共有ファイルシステムである。
- 第三階層ストレージ（HPCI 共用ストレージ） : Gfarm[3]
  HPCI の各センタから利用可能な広域分散ファイルシステムであり、HPCI の複数センタを利用するユーザがセンタ間のデータ共有などのために用いる。

HPCI共用ストレージは「富岳」とは独立して整備されてきたものであり、「富岳」固有のストレージは第一階層ストレージおよび第二階層ストレージである。ストレージの階層化は、高性能かつ大容量なストレージシステムの実現手段として、大規模なスーパーコンピュータで一般的に採用される方法である。「富岳」の前身にあたる「京」でも、「富岳」と同様に二階層のストレージ構成を取っていたが、いずれの階層においてもファイルシステムとしてFEFSが採用されていた。既に述べた通り、「富岳」では第一階層ストレージのファイルシステムとしてFEFSは採用しておらずLLIOを新たに開発したが、その理由は以下の通りである。

- 第一階層ストレージ上のファイル I/O 処理のために利用できるリソースが限られており、その中で安定動作可能な大規模環境向け分散ファイルシステムが存在しなかった
  「富岳」の第一階層ストレージは専用サーバではなく計算ノードに搭載された SSD を用いており、クライアントからのファイル I/O 要求は、当該計算ノードで実行されているサーバが処理する。ファイルシステムを含むシステムに割り当てられた CPU およびメモリリソースは、ノード当たりそれぞれ最大

[1] 富岳は 6 つの FEFS で構成され、そのうち 1 つは全計算ノードにマウントされているため、その FEFS へのアクセスした場合は LLIO を経由しない。4.3.3 でこの理由について述べる。

4 コア、3.2GB（搭載メモリ容量の 10%）であり、専用サーバを用いる場合と比較すると非常に限られている。このようなリソース制限の下において安定動作可能な大規模環境向け分散ファイルシステムは、設計時点では存在しなかった。

- 「富岳」の第一階層ストレージに求められる機能要件を満たすソフトウェアが設計時点で存在しなかった

  「富岳」は「京」の後継にあたるシステムであり、そのストレージシステムは「京」で培った資産の継承と課題を改善するものになっている。「京」の課題を改善するための機能の中には、当該機能を実装しているソフトウェアが存在しないものがあったため、新規に開発する必要があった。その一つとして、「京」で提供していた階層間データ転送機能であるステージング機能の課題を改善するキャッシュ機能が挙げられる。ステージング機能では、ファイルを階層間で転送する場合、ユーザが明示的に転送対象のファイルを指定する必要があったが、ジョブ実行に必要なファイルが指定されていないことによるジョブ実行エラーや、不要なファイルが指定されることによるステージング時間の長期化が、運用上の課題となっていた。「富岳」では、この課題を解決する手段として、第二階層ストレージの FEFS に置かれたファイルに対する透過的なアクセスを可能にし、システムが階層間のデータ転送を自動で実行するキャッシュ機能を採用したが、設計時にはこれを実装したソフトウェアは存在しなかった。

LLIOの機能と成果

前述の通り、「富岳」の第一階層ストレージは、計算ノードに搭載されたSSDで構成され、LLIOは第一階層ストレージ向けに開発されたファイルシステムである。「富岳」では、全計算ノードがSSDを搭載しているわけではなく、16計算ノード毎に1つの計算ノードにSSDが搭載されている。SSDを搭載した計算ノードはSIOと呼ばれ、LLIOのサーバの役割を担い、SIOを含むすべての計算ノードはファイルシステムのクライアントになる。

LLIOは、表 3.2.4に示す三種類の領域をジョブに対して提供しており、ジョブは用途に応じてファイルのパス名を切り替えることでそれぞれの領域を使い分ける。第二階層ストレージのキャッシュ領域（以降、第二階層キャッシュ）は、第二階層ファイルシステム（FEFS）のキャッシュとして動作する領域であり、保存ファイルの入出力はこの領域を介して行う。前述した「京」のステージング機能における課題を改善するキャッシュ機能を提供するのは、本領域である。本領域に加え、LLIOではアプリケーションのI/O最適化を可能にする二種類のテンポラリ領域を提供している。

LLIOでは、SSDの性能をアプリケーションのファイルI/O処理に最大限割り当てるため、データの冗長化を行っていない。そのため、多数のクライアントが実行ファイルなどの共有ファイルに一斉にアクセスした場合、特定のSIOが高負荷状態になり、アプリケーション実行が停滞する懸念がある。LLIOでは、この問題を回避するための機能として、llio_transferと呼ばれる共有ファイルを各SSDへ複製するコマンドを提供している[4]

表 3.2.4 LLIO の提供する三つの領域

領域名	名前空間	用途	性能(File per Process) [5]
第二階層ストレージのキャッシュ	第二階層ファイルシステム透過	保存ファイル	READ: 11.0 TB/s WRITE: 5.97 TB/s
共有テンポラリ	ジョブ内共有	一時ファイル	READ: 22.4 TB/s WRITE: 8.16 TB/s
ノード内テンポラリ	ノード内共有	一時ファイル	READ: 45.2 TB/s WRITE: 18.6 TB/s

表 3.2.4に示す通り、LLIOの各領域は高いスループットを達成している。また、LLIOを用いたアプリケーションの最適化として、「富岳」で重点的に取り組むべき社会的・科学的課題として選定された、ターゲット・アプリケーションの実行時間の短縮に寄与している[6][7]ほか、機械学習ベンチマークの学習データの読み込み高速化に活用されている[8]など、実際のアプリケーションの高速化にも寄与している。

富岳運用における課題と改善案

上記の通り、LLIOは富岳の第一階層ストレージのファイルシステムとして高いスループットを達成し、アプリケーションの高速化にも寄与している。その一方で、富岳現地の運用では課題も発生している。本稿では、それらのうち性能に関するものに焦点をあて、四つの課題について分析を行なった結果について報告し、まとめとして富岳NEXTに向けて継続的に検討、議論されるべき項目を述べる。

課題1. Single Shared Fileの性能スケーラビリティが低い

LLIOはFile Per Processの性能で非常に高いスループットを達成しているが、Single Shared File（SSF）の性能スケーラビリティは限定的である[4]。これは、第二階層キャッシュおよび共有テンポラリ領域の各ファイルのメタデータの一部（ファイルサイズなど）が、特定のSIOで管理（以降、一元管理）されていることに起因する。一元管理では、多数のクライアントがSSFに一斉にwriteした場合、メタデータの更新処理がボトルネックとなり性能がスケールしない。

一元管理とは異なるメタデータの管理方式として、Lustreで採用されている各ファイルのメタデータを分散管理する方法（以降、分散管理）が存在する。分散管理では、write時の明示的なメタデータ更新は行わず、メタデータの参照時にチャンクファイルが格納されている全サーバからチャンクファイルのメタデータを収集し、ファイルサイズなどの情報を復元する。LLIOの設計時に分散管理の採用も検討したが、メタデータ参照時の合計RPC数の増加を懸念し、一元管理を採用した。LLIOはNFS v3のキャッシュコンシステンシーモデル[9]を採用しており、I/O中にもメタデータ参照が発生しうるため、一概にI/O中のRPC数が少ない管理方法がどちらであるかを結論付けることは出来ない。また、このコンシステンシーモデルを採用している限り、単純なクライアントとサーバ間のみの通信では、いずれの管理方法を用いてもクライアント数の増加に比例してサーバ負荷が上昇する。

本課題の改善方法として、クライアント間やサーバ間の通信を行うことも含め、ルーティングの工

夫などを実施することでクライアント数増加によるサーバ負荷上昇を抑えることが考えられる。

課題2. Single Shared Fileの同時アクセスクライアント数に上限がある

LLIOを利用する場合、SSFの同時アクセスクライアント数に上限があり、この上限は富岳の全計算ノード数に満たない。そのため、この上限を超えたSSFにアクセスするジョブは、全計算ノードに直接マウントされたFEFS[2] を利用する必要がある。この理由は、メモリリソースの制限に伴い同時コネクション数に上限があり、この上限を超えたクライアント数のリクエストを安定して処理することが難しいためである。

前述の通り、LLIOは計算ノードで構成され、LLIOを含むシステムが利用できるメモリ容量は、計算ノード当たり3.2GiB（搭載メモリの10%）である。システムのメモリ使用量をこの制限内に抑えるために、LLIOでは各計算ノードのコネクション数に上限を設定しており、新たなコネクション確立を行う際にコネクション数の上限を達していた場合は、古いコネクションを切断し新たなコネクションを確立する。課題1.で述べた通り、SSFでは多数のクライアントが特定のSIOに対して一斉にリクエストを発行する可能性があるため、コネクション要求も一斉に当該SIOに発行される可能性がある。このような状況下では、一つのクライアントがSIOと確立したコネクションが、他クライアントのコネクション確立のために強制的にコネクションが切断されるということが繰り返され、一向にリクエスト処理が進まなくなる可能性がある。安定したリクエスト処理を行うためには、同時アクセス可能なクライアント数の上限を設定しておくことが必要と判断し、LLIOでは、コネクション数の上限をSSFに同時アクセス可能なクライアント数の上限としている。

計算ノードのメモリ容量を増やすことによる一定の改善は見込まれるが、10万規模のコネクションを常に維持するためには相当のメモリ容量が必要になるため、課題1.の改善案で挙げたような、ルーティングの工夫などを実施することで、クライアント数が増えてもメモリ使用量が増えないような対策が必要と考える。

課題3. メタデータアクセス性能がFEFS直接アクセス時より低い

LLIOの第二階層キャッシュおよび共有テンポラリ領域の名前空間はFEFSのMDSで管理しているが、これらの領域のファイル作成や削除の性能は、FEFSに直接アクセスする場合と比較すると低い[3]。この理由は、LLIOを利用する場合、FEFSに直接アクセスする場合と比べ、SIOを追加で経由する分、経由ノードが 1 ノード多いためと考えられる。

名前空間に関する操作はSIOを経由せずFEFSに直接アクセスする方法も考えられるが、システムの安定性観点ではFEFSのクライアント数を減らすことが望ましいため、1ボリュームを除いてSIOのみにFEFSをマウントし、各計算ノードはSIOを経由してFEFSにアクセスする方式を選択した。また、名前空間の操作を一時的にSIOにバッファリングしておき、アプリケーションの実行とは非同期にFEFSに反映する方法も考えられた。しかし、このような方式では、アプリケーションがLLIO経由でファイルを作成した後に、ユーザが直接FEFSにアクセスし同一パス名のファイルを作成した場合、アプリケーションの後続のファイルアクセスがエラーになるため、ファイルシステムの利便性という観点で同期的にFEFSの名前空間の操作を行う方式を採用している。

[2] 富岳には 6 つの FEFS が存在するがそのうち 1 つは全計算ノードにマウントされており計算ノードから直接アクセスが可能で、残りの FEFS は SIO にのみマウントされており LLIO 経由でのアクセスが前提となっている。

富岳や富岳NEXTを利用するユーザは以前にも増して多様化し、標準仕様を求めるユーザ、性能を出すためにシステム固有の仕様を理解して使いこなすユーザなどニーズも多様化していくと想定される。特定の仕様で全てのニーズを満たすことは難しいことを認識した上で、提供するシステムの仕様がどのユーザ向けのものなのか、当該仕様で満たすことができないユーザのニーズはどのように満たしていくのかの検討が必要と考える。

課題4. llio_transfer指定漏れによるSIO高負荷発生

llio_transferは、複製対象ファイルにアクセスするクライアント数が多くなるほど、より大きな効果を持ち、大規模ジョブ実行時には、利用がほぼ必須となっている。しかしながら、llio_transferはLLIOが提供している固有のコマンドのため、大規模ジョブ実行ユーザが指定を忘れてしまうことがある。その場合、特定のSIOに負荷が集中することによってread処理のスローダウンが発生し、ジョブ実行時間経過でジョブが中断されるなどの現象が発生する。

環境固有の必須処理に関しては、ユーザに明示的に実行させるのではなく、システムによって自動的に行われることが利便性の観点では望ましい。本課題の場合、共有ファイルのシステムによる自動検出と複製実行が考えられる。

以上、富岳の運用で上がっているLLIOの性能関連の課題それぞれについて、その原因や改善案を述べた。本報告で分析した課題から得られた富岳NEXTに向けた検討項目は、大規模ジョブ実行時の負荷軽減と多様なユーザを考慮した仕様策定である。

一点目について、本稿ではリクエストの集中やコネクション数の増大を負荷上昇の原因として上げ、ルーティングの工夫などによる改善が必要であると述べた。データ科学が以前にも増して注目される中で富岳NEXTでは、ファイルI/O性能に対する要求も以前より厳しくなるものと推測される。今回述べた負荷軽減に留まらず、アプリケーションからのI/O性能要求を踏まえた性能向上も必要になる。

二点目について、本稿では、環境固有処理のシステムによる自動実行（ユーザに対する処理の隠蔽）と多様なニーズを意識した仕様の策定を挙げた。後者については、単一の仕様で全てのユーザのニーズを満たすことは困難であり、また、多様なニーズに合わせた多様な機能および仕様を全て新規に開発することも困難と考える。既存ソフトウェアの活用も視野に入れつつ、仮に新規のソフトウェア開発を行うのであれば、当該ソフトウェアで満たすニーズ、また、満たされないニーズとそれを満たす方法の明確にしていくことが必要と考える。

【参考文献】

[1] スーパーコンピュータ「富岳」におけるファイルシステム・電力管理の機能強化, https://www.fujitsu.com/jp/documents/about/resources/publications/technicalreview/2020-03/article05.pdf（2023年2月参照）

[2] スーパーコンピュータ「京」の 高性能・高信頼ファイルシステム, https://www.fujitsu.com/downloads/JP/archive/imgjp/jmag/vol63-3/paper08.pdf（2023年2月参照）

[3] HPCI共用ストレージ概要・構成, https://www.hpci-office.jp/info/pages/viewpage.action?pageId=111380786（2023年3月参照）

[4] スーパーコンピュータ「富岳」のファイルシステム, https://www.jlug.info/_files/ugd/8d1181_f4c8a5e4774940d09674d175de5def4b.

pdf（2023年2月参照）
[5] LLIO/FEFS for the supercomputer Fugaku, https://8d118135-f68b-475d-9b6d-ef84c0db1e71.usrfiles.com/ugd/8d1181_e7eca27010b14000890630c59086138a.pdf（2023/4月参照）
[6] 鈴木 惣一朗, 伊東 聰, 酒井 憲一郎, 稲田 由江, 三吉 郁夫, 石川 裕, 宮野 悟, ヒト全ゲノム解析プログラム Genomon のスーパーコンピュータ「富岳」向け最適化と性能評価, 情報処理学会研究報告、Vol. 2021-HPC-178, No. 18, pp. 1–9 （2021）
[7] 理化学研究所 計算科学研究センター, 富岳コデザイン・レポート ～フラッグシップ 2020 プロジェクト・テクニカルレポート～, https://www.r-ccs.riken.jp/wp/wp-content/uploads/2022/03/fs2020-report-j.pdf（2023年2月参照）
[8] World's Largest Scale Deep Learning on Supercomputer Fugaku Achieved World's Highest Performance, https://blog.fltech.dev/entry/2021/12/09/mlperfhpcv1-fugaku-en（2023年2月参照）
[9] NFS Version 3 Protocol Specification. https://www.ietf.org/rfc/rfc1813.txt（2023年2月参照）

3.2.4.3.4　次期ストレージシステムに対する考察

次期ストレージシステムの性能については、次期スーパーコンピュータの演算性能、メモリサイズ、ノード数などとのバランスがとれていることが重要である。一度に発生するデータアクセスはメモリサイズが上限である。グランドチャレンジ時など全システムを用いる場合は全プロセスからのアクセスが一度に発生するため、ストレージシステムはそのアクセスに耐えられる必要がある。ストレージシステムのバンド幅については高速であるに越したことはない。その際、FPP、SSFなどのアクセスパターンについて充分なバンド幅が求められる。

次期スーパーコンピュータの演算性能を20 EFlops、メモリ容量を20 PB、ノード数を数千～十数万と仮定する。この時、20 PBのデータを1分でアクセスできるように設計すると、ストレージシステムのバンド幅は350 TB/sが要求される。またメモリサイズが20 PBであるため、その30倍の容量を保持すると600 PB必要となる。ノード数が数千～十数万であるため、各ノードに数十～数百のプロセスが起動するとして秒間1億操作（100M IOPS）程度のメタデータ性能が必要となる。ここでの操作はファイル作成、ファイルオープンなどの操作である。さらには、600 PBの容量では不足するため、その10倍位（6 EB）のどこからでもアクセス可能な広域のストレージシステムが必要である。

動向調査により、2028年～2030年においては、PCIe Gen7の製品の出荷が始まることが予想される。フラッグシップシステムにおいては最先端の技術をいち早く導入して、上記の要求を満たすことが可能かどうか検討することが望まれる。PCIe Gen7は16レーン利用すると1.6 Tbps（＝200 GB/s）の性能となる。ストレージに対しては4レーンを用いて最大50 GB/sとなる。一方で、動向調査をもとにHDDについては50 TB、600 MB/s、10W、SSDについては240 TB、32 GB/s、40Wを仮定する。

350 TB/sを達成するためには、HDDでは58万台必要となるが、その時の消費電力は5.8 MWとなってしまい現実的ではない。SSDの場合は1.1万台ほどとなる。容量は600 PBであるが、冗長性を持たせる必要があるため実容量としては96D9PのWide LRC (Locally Recoverable Codes) を用いると657 PB

であり、各SSDの容量は60 TBでよいこととなる。ストレージサーバを1.6 Tbpsの4レーンで接続すると800 GB/sのネットワーク帯域となるため、440サーバほど必要となる。このとき、各サーバには60 TBのSSDを26台搭載することとなる。ここまでは物理的に350 TB/sを達成するためのハードウェアの構成を考えたが、これだけでは十分ではなく、さらに、数百万プロセスからの100M IOPSのメタデータ性能、FPP、SSFにおける350 TB/sの安定した高いバンド幅を実現するためのストレージシステムの構成が求められる。ファイル数が多くなった時にも性能低下が起こらないようにすることも必要である。つまり、物理的なバンド幅を達成するためのハードウェア構成だけではなく、必要なメタデータ性能、FPP、SSFにおける高い性能を達成するためのハードウェア構成とストレージシステムを同時に考える必要がある。

別の解として、350 TB/s、600 PBのストレージシステムを階層的に実現することも考えられる。これは富岳でも用いられている方式であり、先の案に比べハードウェアコストが下がると考えられる。一方で、現在富岳で起きている様々なストレージの問題の解決も求められる。例えば、1次ストレージは350 TB/sは実現するものの容量はメモリの倍の40 PBとし、2次ストレージではバンド幅は20 PBのデータを5分でアクセスできるよう70 TB/sとし、容量は600 PBとする。このとき、1次ストレージに対するバンド幅要件は先と同じであるが、各SSDの容量は3.7 TBでよくなる。1次ストレージは2次ストレージへフラッシュすることを前提とすると冗長性は必須ではないと考えられる。SSDよりも高速だが容量が少ない不揮発性デバイスが利用可能であればその選択肢もありうる。また、1次ストレージでは100M IOPSのメタデータ性能だけではなく、FPP、SSFにおける350 TB/sの高い性能が求められる。2次ストレージにおいて、70 TB/s、600 PBを実現するためには、冗長性を考慮しても 240 TB、32 GB/sのSSDが2,750台あればよい。先のストレージサーバであれば106サーバほどで実現可能である。HDDの場合は12万台必要となり、消費電力は1.2 MWとなるため現実的ではない。2次ストレージは計算ノードからの直接アクセスは行わず、1次ストレージとの間でのデータ転送だけであると考えると、メタデータ性能は100M IOPSも必要はなく10M IOPS程でいいと思われる。ただし、この階層ストレージに対し、必要なストレージ性能を達成するためのハードウェア構成とストレージシステムを同時に考える必要がある。

広域ストレージシステムについては、6 EBを実現するために、HDDでは12万台、1.2 MW、SSDでは2.5万台、1 MWとなる。バイト単価がHDDの方が一桁よいことを想定するとHDDを用いるのがよいと考えられる。

3.2.4.4 来年度計画

今後、他のストレージアーキテクチャについての調査を進めるとともに、次期ストレージシステムの要求を満たすストレージアーキテクチャの提示を行う予定である。

3.2.5 数値ライブラリ

3.2.5.1 調査目的

数値計算ライブラリは、科学技術アプリケーションプログラムを中心として共通に使われる、典型的な数値アルゴリズム群をコード化しライブラリの形にまとめたものである。高度な数学やプログラミング技術が投入され実現されており、通常、利用者が短期間に作成することは困難である。利用者からは数値計算ライブラリが提供する高性能かつ高信頼性を根拠に、過去現在そして将来にわたって継続的な利用が期待されている。数学的な分類として各種存在するが、数値計算の最も基本的な機能として以下の４手法は必須である。

- 線形方程式（連立一次方程式）

- 固有値問題（特異値計算も含む）
- 高速フーリエ変換
- 乱数生成

数値計算ライブラリがアプリケーションプログラムの中核的な部分に使われていることから、その性能がアプリケーションに大きく影響する。性能の指標はさまざま存在するが、例えば実行時間（速度）、消費電力、計算精度などがある。また、大規模な計算では並列化の程度によって、利用する計算機アーキテクチャや問題サイズへの制限、並列化効率にも影響する。さらに、昨今では、旧来のFortranだけではなくC、C++に加えてPythonやJuliaなどの新言語から呼び出し可能かなど言語バインディングの問題もある。さらに、バックエンドのランタイムモデルとの親和性を考慮した設計になっているかなど、最新のプログラミング言語との連携がなされているかの観点、ひいては、保守性・継続性の面からも調査が必要である。

3.2.5.2 調査内容

本SWGでは、2028年頃に登場する次世代スパコンシステムにおいて、現在の「富岳」級システムでの動向調査を進め、その延長線上の技術外挿ならびにいくつかの技術動向予測を加え、次世代スパコンにおける数値計算ライブラリの様相についてまとめる。

本中間報告書では、2022年8月に発足以降進めてきた、「富岳」開発者としての数値計算ライブラリ開発時の振り返りを通じて、現状の問題点や次期システムへの引継ぎ点をまとめる。次に、既存の数値計算ライブラリの動向調査を中心として、各種ライブラリに求められる機能とその対応状況など、様々な観点からの比較検討の中間報告をまとめる。最後に、残り2年間に行うべき継続調査項目についてまとめる。

3.2.5.3 調査結果

3.2.5.3.1 「富岳」振り返り

3.2.5.3.1.1 富士通視点での富岳の振り返り

以下に設計時方針と達成状況の概要と、各数値計算分野における機能・性能および課題をまとめる。

設計時の方針と達成状況

ハードウェア性能を引き出すことに重点を置く設計時の方針に沿い、京からの継続である既存ライブラリ機能の移植およびチューニングを行った。特に数値計算ライブラリの基本機能であるBLASおよびFFT機能群に対しては、新規で特徴的なSVE命令を十分に活用してA64FX向け高性能ルーチンを開発し性能目標をクリアした。それらの基本機能を利用して構成されるLAPACKなどの上層機能群も高速化され、さらに新しいトレンドへの取り組みとしてPython(NumPy/SciPy)でのチューニング版BLAS/LAPACK結合にも対応した。開発当初の狙い通りに、ハードウェア性能を引き出して過去の資産を継承する開発が達成されたと考えられる。機能と性能についての具体的な取り組みを次節に記載する。

機能・性能

密行列線形計算分野に関しては、基本的行列演算機能BLASのSVE対応によりArm Performance LibrariesやOpenBLASに先行してA64FX向け高性能ルーチンを提供できた。LAPACK機能群に対してはスレッド並列化をBLAS内だけでなく上層でも並列化も行っているが、Intel MKLと比較して特に固有値ソルバの性能が大きく劣っていることが判明している。スレッド並列

性能の改善に関しては、当初はタスク並列を用いたPLASMAライブラリをサポートすることで高性能なルーチンを提供できる可能性があると考えていたが、コミュニティ側での開発は停滞気味で想定よりも性能改善と普及が進まなかった。

高速フーリエ変換分野に関しては、富士通製のSSL IIライブラリ既存機能の移植およびチューニング（SVE化、A64FX対応）を行った。また、デファクトスタンダードとなりつつあるFFTW3ライブラリにSVE対応のチューニングも行った。Intel Xeon上でのFFTW3に性能は及ばないが、FFTカーネル内の演算の間に依存関係の長い連鎖があることにより命令レイテンシの影響を受けやすいためであり、ハードウェア要因が大きいと分析している。

その他の疎行列線形計算、疑似乱数生成、常微分方程式などの分野に関しては、SSL IIライブラリ既存機能の範囲で移植ルーチンを提供した。

新しいトレンドへの取り組みとしてはPython(NumPy/SciPy)とチューニング版BLAS/LAPACKの結合に加えて、AI向けのライブラリ(FP16版BLAS、oneDNN)は要望が上がって開発の後半で対応して富岳で必要とされる範囲の機能・性能を実現できた。

使用上の課題

アプリケーションのポータビリティが重要視される傾向が強まる中で他ライブラリとの互換性の無い独自APIでの機能提供では利用されにくいという課題が見えてきている。

さらに、GCC環境での利用要望も上がっているが、SSL IIのBLAS、LAPACKはGCC環境では利用できない。これは、富士通独自のハードウェア機構を使用して性能向上させるために、富士通コンパイラを使用してセクタキャッシュ制御機能を含むランタイムを結合することを前提としているためであるということも課題である。

3.2.5.3.1.2 理研視点での富岳の振り返り

「富岳」開発時点でグランドチャレンジユーザとのヒアリングから、密行列計算の中で固有値ソルバと行列積の需要が高く、さらに高速化の要望が強いことが判明していた。富士通側は標準的なNetlibソフトウェアのA64FXへの移植に注力をし、理研はスケーラビリティや高性能化に対応するべく分散並列版の固有値ソルバEigenExaと分散並列版の行列行列積ルーチンPDGEMMの2.5次元アルゴリズム版SC-SUMMA-25Dを開発した。それぞれの開発に対して要約を以下に示す。

EigenExa

EigenExaは国内で開発されるの数少ない競争力のある密固有値計算ライブラリの一つである。「京」からの継続開発となり、新しいハードウェア(A64FX、TofuD)とソフトウェア(コンパイラ、MPI、富士通SSL II BLAS)への対応が主な移植ポイントであったが、分割統治法において「京」版ではスケーラビリティに問題がある実装方式を採用していたため、「富岳」版ではデータ分散方式を動的に変更しながらシステムの利用効率を最大化することで性能劣化の要因を改善する実装を行った。しかしながら、新実装にはMPIのサブコミュニケータの動的生成が必要であり、当該コストを削減するための独自のサブグループ管理方式などの追加開発コストを要したため、EigenExaの安定版の公開は2022年10月までかかってしまった。しかしながら、性能スケーラビリティは8000ノード規模まで確認され100万次元の対角化を「京」と同程度の計算ノード数を用いて1時間以内で行う目標は達成している(https://www.r-ccs.riken.jp/labs/lpnctrt/projects/eigenexa/)。

2.5次元分散並列行列積ルーチン

SC-SUMMA-25Dは2.5次元アルゴリズムを採用した分散密行列積の実装である。2.5次元アルゴリズムそのもののアイデアは明確であるため機能を制限したプロトタイプは「京」上でも動作し、有効な性能向上と高いスケーラビリティを確認している。「富岳」においても機能制限版は問題なく動作し2021年4月にα版をリリースしている(https://www.r-ccs.riken.jp/labs/lpnctrt/projects/25dpdgemm/)。

その他

「富岳」開発の後半から新しいトレンドとしてAI分野への取り組みが必要となり、理研担当範囲を拡大しBatchedBLASに対応する作業を実施し、富士通 SSL II BLASが提供するすべてのAPIに対してBatchedAPIを実現した。当該機能はAIフレームワークにも採用されたが、AI分野で利用される行列サイズが小規模であることとイレギュラー(極端な非正方)であることから期待するほどの性能が得られないなど現時点でも重要な改善点が残っている。

3.2.5.3.2　数値計算ライブラリ動向調査

本調査研究の過程で、米国エネルギー省関係（DOE）の国立研究所のメンバと打ち合わせにおいて、エクサスケール計算機プロジェクト(ECP)において開発を進めてきた数値計算ライブラリの動向調査もすすめた。その中で、オークリッジ国立研究所のKeita Teranishi氏から、米国においても数値ライブラリの重要性は高く、特にプログラミング言語やアプリケーションの長期活用に伴い継続性(sustainability)がキーワードとなるとの見解を示され当SWGでも同様の立場であることを共有した。本調査研究でも同様の指標のもとに国内外での開発体制や2028年ごろのソフトウェアの動向を調査している。本質的には目的は同様であるが、ECPでは多くのソフトウェア群を開発してきており、日本国内の状況とはやや異なっている。特定目的のソフトウェアには複数のソフトウェア実装の選択肢を可能としてきており、必須ソフトウェアの継続的な利用が途切れないよう戦略的開発を進めている。

数値計算ライブラリは基礎数学的なものから応用指向の特定問題までをカバーするソフトウェアカテゴリである。DOE関係者との議論からECP関連ソフトウェア開発時に用いられた分類を基に、本調査研究の目的を達成するためにいくつか追加したものを以下に示す。大まかに、数値計算を意図する様々な機能とソフトウェアが階層的に列挙されている（数字が大きいものほど少ないものを内部で参照する関係もしくは数学的な抽象度または量的な複雑度が高い）。

1. Kernels: 基本演算ならびにデータ管理等
2. Dense linear algebra: 密線形代数
3. Sparse iterative solvers: 疎反復ソルバ
4. Sparse direct solvers and hierarchical matrix solvers: 疎直接ソルバもしくは階層行列ソルバ
5. FFT: 高速フーリエ変換
6. Frameworks: プログラミングフレームワーク
7. Mesh: 格子生成
8. Graph: グラフ・ネットワーク
9. DSL: Domain-Specific Language

10. Tensors, convolution, decomposition, distributed format: テンソル(高次元データ科学向けデータ)関係

なお、項目9のDSLは数値計算ライブラリを調査するうえでは、数値ライブラリに分類し調査対象とすべきとはいいがたいが、元の分類を尊重して残している。DSLは、当調査研究では言語やシステム・ソフトウェアとの関連において調査が進められており、本サブグループでは本格的な調査を行わない。
また我々はこれら10カテゴリに加えて、「京」・「富岳」の利用者からの要求が高い

11. Pseudo Random Number Generator: 疑似乱数生成器

を加える。
以下、階層ごとに主要ライブラリをあげ、中間報告作成時までに判明している諸元や機能特性などの調査結果をまとめる。

3.2.5.3.2.1 Kernel(カーネル)ならびにそれらを構成する基本機能

数値計算に特化した基本演算やデータ群を自動生成するツールもしくは実行時に動的管理する最下層ライブラリ群を指す。 主に、数値計算ライブラリを下支えする基本的な構造とデータ群さらには高性能な数値計算ライブラリを効率的に構築するためのツールも含めて、数値計算ライブラリを利用する上位の視点から開発者がもっぱら専門性を持って利用する機能を有した最下層の基本ツールとして分類する。ただし、BLAS等の数値線形代数の基本部分はより上位の分類をする。

数値計算ライブラリ構築に必要な機能として

(ア) 基本となる数値データ形式、ベクトルや行列データの独自管理
(イ) メモリや分散環境下での通信回りやファイル形式までの管理
(ウ) 一般のプログラミング言語のみでは対応しない、線形代数やその他数値計算の基本カーネルを自動生成する独自の数値計算カーネル生成機能（ソフトウェアとしては数値ライブラリ開発向けフレームワークや DSL として分類も可能）

などを想定する。なお、ここで想定するカーネル機能は多くの数値計算ライブラリでは独自機能として実装されることもあり、いくつかの上位ライブラリの中に含まれていることが多い。我々が分類するいくつかの階層に重複して登場するものがある。 また、システム・ソフトウェアとして一般的に認知されている並列化テンプレートライブラリIntelTBB、Thrustや実行時ランタイム、通信ライブラリはこの中には含めていない。しかしながら、数値ライブラリはこれらとの強い相関があるため、継続して関連を調査する必要がある。

本カテゴリは基本機能を提供する最下層であり、多くが論文としてその概念と設計思想がまとめられている。 また、直接の数値計算機能を提供するものではないが、カーネル部分を管理し自動生成するものについても開発や品質維持の観点から重要視し調査する必要がある。

プログラミングならびに基本データ管理等：

- Kokkos Kernels: Kokkos ライブラリにおいて各種データや実行管理をつかさどる C++の基本クラスライブラリ群である。 数値計算にとどまらず、一般のプログラミング言語として分類しても構わないほどの機能を提供する。参照論文 Kokkos Kernels: Performance Portable Sparse/Dense Linear Algebra and Graph Kernels, SAND2021-3421 O (https://doi.org/10.48550/arXiv.2103.11991), コード配布元: https://github.com/kokkos/kokkos-kernels,

- Ginkgo: 疎行列計算のための各種線形作用素を抽象化し、疎行列開発のための行列データハンドリング、各種デバイスへの拡張機能、さらには C++の柔軟なクラス機能を活かしつつ拡張可能ソルバ構築を目指し設計された近代的な C++数学ライブラリの一つである。疎行列ソルバとしての一般的ユーザには機能提供されるため疎行列ライブラリでも紹介する。参照論文 Ginkgo: A Modern Linear Operator Algebra Framework for High Performance Computing, ACM Transactions on Mathematical Software Volume 48, Issue 1,Article No.: 2, pp 1–33 (https://doi.org/10.1145/3480935), コード配布元: https://github.com/ginkgo-project/ginkgo
- PETSc: 標準的な偏微分方程式求解のためのフレームワークの一部として疎行列ソルバが提供されているが、その細部を分析すると分散環境下における数値線形代数を実現可能な基本機能を有している。C 言語や古典的な Fortran からの利用であるがその機能は拡張性にはやや難点があるが老舗として十分な機能提供をしている。(https://petsc.org/release/)
- Eigen: C++プログラミング環境下で線形代数演算を比較的容易に実現するためのライブラリ。近代的な C++実装とは言えないものの、初期の実装としては利用者によるデータや型機能拡張の手段が提供されており当カテゴリに分類している。コード配布元: https://eigen.tuxfamily.org/

代表的な高精度データ型と拡張機能：

- GMP、MPFR、libquadmath、QD などの多倍長計算ライブラリは 4 倍精度以上の高精度計算を支える有用なツールでもある。(GMP: https://gmplib.org/, MPFR: https://www.mpfr.org/, GCC-Quad-Precision Math library: https://gcc.gnu.org/onlinedocs/libquadmath/, QD: https://www.davidhbailey.com/dhbsoftware/)
- 区間演算を支援する Intlab は Matlab で利用されるが、同様の機能を提供する kv ライブラリ(http://verifiedby.me/kv/)なども存在する。C++とともにデータと演算をそれぞれ個別に実装する環境が整うとともに、今後同様の機能が高性能計算分野で本格的に利用される可能性がある。

数値計算カーネル作成のための有用なツール群

- SPIRAL: 演算子言語（OL）によって定義された計算アルゴリズムの自動的な等価変換やカーネル合成により最適化されたカーネルを生成する仕組み。主に FFT カーネル生成に応用されている。参照論文 SPIRAL: Extreme Performance Portability, IEEE From High Level Specification to High Performance Code, Vol. 206, No. 11, 2018 (https://doi.org/10.1109/JPROC.2018.2873289)、コード配布元: https://github.com/spiral-software/spiral-software
- CUTLASS: 行列積に特化しているが最適化された CUDA 向け GEMM カーネルを生成するテンプレート群である。最新の CUTLASS3 では内部でテンソル構造を CuTe と呼ばれる再帰的ライブラリにより抽象化して扱うことができる。コード配布元 https://github.com/NVIDIA/cutlass
- ATLAS、OpenBLAS、BLIS: BLAS 実装系であるが、アーキテクチャ仕様などからアセンブラコードを自動生成する開発フレームワークでもある。特に、ATLAS(https://math-atlas.sourceforge.net/)が起源といわれているパラメータ自動チューニング機能により最終的

なコード選択の手順を機械化することが先駆的に実施されている。OpenBLAS は Goto 氏による GotoBLAS の進化系でありコミュニティの支援より各種アーキテクチャに対応している (https://www.openblas.net/)。BLIS も同様に高性能 BLAS コード生成の自動生成フレームワーク機能を提供する(https://github.com/flame/blis)。近年では AMD プロセッサ向けの BLAS として推奨されたり、A64FX で最速となるカーネル生成に成功するレポートが出るなどコミュニティ内での支持を広げている。

- Libxsmm: 小規模または疎行列の特殊な形状の行列積に対する最適化コード生成を実現するライブラリである。コンパイラとは独立した JIT 最適化に基づいている。インテルアーキテクチャに特化し、FP64、FP32、BF16、int16、int8 などのデータ型がサポートされている。コード配布元: https://github.com/libxsmm/libxsmm

この他にも数値計算には特化していないが、富岳のAIフレームワーク中で高速化に寄与したXBYAK(マシン語生成のためのC++クラスライブラリ)も存在する。これら以外にもコード生成フレームワークは他にも数多く存在するため、最終報告書完成までにより多くのレビューを継続実施し詳細をコンパクトにまとめたい。

3.2.5.3.2.2 Dense linear algebra(密線形代数)

ベクトルと行列に関する和や積などの基本演算から、それらを用いて連立一次方程式や固有値・特異値など科学技術計算に必要となる線形代数計算のコア部分を提供する。本カテゴリでは特に行列データが密（非ゼロを区別せずすべてのデータを保持する形式）である場合を指す。基本的な概念や機能はBLASやLAPACKに集約されるが、対象となるプラットフォームやプログラミング言語・モデルによって同一機能でも異なる実装やベンダによる最適化ライブラリが存在する。

機能分類による代表的ライブラリ

【BLAS】: 密行列とベクトルに関する基本演算を中心に実数もしくは複素数を体とする線形演算操作をサブプログラム化し集約したライブラリである。参照実装であるNetlib版BLASがFORTRANで書かれているため、FORTRANでの利用が想定されるが、近年ではC/C++で作成されることが多く（内部的にはアセンブラやインラインアセンブラやイントリンシック命令を使う）、各種言語への方言や拡張がなされて利用可能である。

- Netlib 版 BLAS: BLAS の機能を参照実装により定義するライブラリ (https://netlib.org/blas/)
- OpenBLAS: GotoBLAS として知られていたものが継続され整備されたものである。各種プロセッサに最適化された BLAS を自動生成可能な実装の一つである (https://www.openblas.net/)
- MKL 版 BLAS: Intel 社が x86 プロセッサ向けに最適化を施した実装系である。中間報告書を執筆時点で利用可能な環境として Intel Compiler(2020 年版まで)に付属するもの、oneAPI に付属するものの 2 系統が存在する。後者は oneDNN を意識した実装系が含まれるなどの違いが存在する。
- cuBLAS: NVIDIA 社の CUDA SDK に含まれる NVIDIA の CUD 向けの BLAS 実装である。1GPU 上のシングル CUDA ストリームで動作する BLAS 用である。
- rocBLAS: AMD 社の RADEON などの実行環境である ROCm 上で動作する BLAS である (https://github.com/ROCmSoftwarePlatform/rocBLAS)

- BLIS: 各種プロセッサに最適化された BLAS を自動生成可能な実装系の一つである (https://github.com/flame/blis)
- SSL II 版 BLAS: 富士通が開発する SSL II に含まれる BLAS の実装。富士通スパコンに搭載されるプロセッサ系列によって実装が存在する。「富岳」に搭載される A64FX に対応するのは V4.0L20.
- ArmPL(Arm Performance Library)版 BLAS: Arm 社が arm プロセッサ向けに最適化した実装系である。BLAS とともに後述する FFT など数学関数を含んでいる。

【extended BLAS】: BLASに精度やその他の機能拡張したもの、BLASのサブセットに特殊な実装を加えたものをここでは分類して取り上げている。

- Netlib 版 XBLAS: BLAS のルーチンに対して被演算子のデータ精度と算術演算の精度を分離したインターフェースを持ち、算術演算はデータ精度より高い精度（最大で double-double 型 4 倍精度）に対応した混合精度対応の拡張精度 BLAS である。
  m4 マクロを使用したやや古典的な実装方式である(https://netlib.org/xblas/)
- MPBLAS: MultiPrecision の線形計算パッケージの先陣的な実装である MPLAPACK で提供される BLAS である。GMP、MPFR、QD などの高精度浮動小数点ライブラリが提供する浮動小数点データが利用可能である
- (https://github.com/nakatamaho/mplapack)。
- cuBLASLt: cuBLAS の GEMM に限定した軽量実装版である。
- cuBLASXt: 複数 GPU で動作する cuBLAS の拡張実装である。
- cuBLAS ならびに rocBLAS の Ex ルーチン: いずれも混合精度インターフェースを提供するものである。GPU が提供する特殊アキュムレータを活用した内部実装的な特徴があり、blockFMA の概念（アキュムレータが高精度）を説明して blockFMA に対応したルーチンを提供する。

【LAPACK】: 密行列の連立一次方程式ソルバ、固有値ソルバ、特異値ソルバなど線形計算の重要なカーネル部分のBLASよりも上位部分を収録したライブラリである。BLASと同様の実装方式であるが、現在でも比較的頻繁な機能拡張や更新がなされている。

- Netlib 版 LAPACK: LAPACK の仕様を定めている参照実装である。ただし、ソースコード上はシングルスレッドモデルを想定しているため、並列化やその他のハードウェア拡張は各ベンダや開発者に委ねられている(https://netlib.org/lapack/)。
- MKL 版 LAPACK: MKL BLAS と同様に、Intel 社が提供する MKL に含まれる LAPACK 部分を指す。スレッド並列化や NUMA 制御など共有メモリ環境での独自拡張と OpenMP との連携などがなされている。独自の拡張 API や拡張環境変数を用いることでスレッド実行やアフィニティ関連について詳細な指定が可能である。
- SSL II 版 LAPACK: 富士通が開発する SSL II に含まれる LAPACK の実装部分を指す。OpenMP でのパラレルリージョン内外のスレッド並列動作方式を切り分けに対応するとともに、スレッドグループの入れ子構造の機能も提供している。

【ScaLAPACK:】: 分散並列版のLAPACK機能とほぼ同等の数値アルゴリズムを実装し集約したライブラリである。分散データやプロセス管理にBLACS、分散版のBLASに相当するPBLAS、逐次的な数値計算カーネルにはBLASとLAPACKを利用するなど、各階層の数値計算ライブラリが組み合わされた実装となっている。

- Netlib 版 ScaLAPACK: 仕様を定めている参照実装である。(https://netlib.org/scalapack/)。
- MKL 版 ScaLAPACK: MKL に含まれる ScaLAPACK 部分である。
- SSL II 版 ScaLAPACK: SSL II の提供する ScaLAPACK 実装である。ilp64(64 ビット整数によるインデックス指定)ができないなど他の実装と比べ制約がある。

【その他】

- SLATE: ECP での新規開発物の一つである。ScaLAPACK や LAPACK の機能を C++の新しい実装によって、より柔軟なデータ構造や最適化、ハードウェアへの対応などを実現するものである。ScaLAPACK と同様各階層に対応する数値ライブラリも C++化されている(BLAS++、LAPACK++など)(https://icl.bitbucket.io/slate/)。
- PLASMA: メニイコアアーキテクチャ上でスレッド並列計算モデルに基づいた実行により近代的なメニーコア環境での高性能線形代数計算をサポートする数値計算ライブラリである。初期は QUARK と呼ばれるタスク管理機構により実装されていたが 2016 年ごろに OpenMP のタスクモデルに移行している。また、タスク並列でのデータの局所性を高めるために整合寸法配列を使用する LDA 方式からタイル管理方式に移ったのも PLASMA からである(https://icl.bitbucket.io/plasma/)。
- MAGMA: GPU アーキテクチャならびにタスクベースモデルに基づかない部分の実装を担当する数値計算ライブラリである。BLAS 相当の機能を発揮する MAGMABLAS や最新版では疎行列向けの実装や DNN(Deep Neural Network)向けに特化した実装 MagmaDNN も公開している(https://icl.utk.edu/magma/)。
- EigenExa: 理研が開発する密行列用並列固有値ソルバである。「京」時代から開発が継続されており、大規模な第一原理計算シミュレーションからデータ同化を用いた気象予報にまで利用されている(https://www.r-ccs.riken.jp/labs/lpnctrt/projects/eigenexa/index.html)。
- ELPA: ドイツの研究者グループにより開発された密行列用並列固有値ソルバである。欧米でのスパコンセンタでの動作実績がいくつか報告されている。Modern Fortran の機能を使って開発が進められており、C++に移行する各種ライブラリとは一線を画す存在である。

【バッチ機能】

メニーコア環境もしくはGPU環境において、大量の小規模問題を同時に解くバッチ処理の形態に対応するBatchedBLASもしくはその上位のBatchedソルバ機能が提供され始めている。Batched処理に特化したライブラリは理研が開発するBatchedBLAS generatorにより生成されたライブラリや固有値ソルバEigenG-Batchedが代表的である。その例外は、BLASの拡張機能として、MKL版BLAS、ArmPL版BLAS、cuBLASなどに収録されている。また、Batchedソルバはcusolverに線形ソルバ、固有値ソルバ、特異値ソルバの機能が提供されている。

3.2.5.3.2.3 Sparse iterative solvers(疎行列向け反復ソルバ)

偏微分方程式やグラフ解析などに登場する疎行列は行列要素にゼロが多く含まれるものを指し、通常は非ゼロ値のみを保持した圧縮形式で保持することが通常である。また、行列サイズもしくは次元が数千万以上になると反復法と呼ばれる解法が利用される。反復法は残差ベクトルや誤差ベクトルから作られる目的関数に対する勾配方向や共役方向での最小値探索などの方法で目的関数が基準値以下になるまで反復を繰り返す解法である。 近代的なライブラリのほとんどがクリロフ部分空間法を採用しており、行列ベクトル積、内積、2ノルムの計算ができればよく、ほとんどのライブラリがこれら操作を部品化して実装し、反復アルゴリズムを構成している。また反復計算の収束を加速する前処理行列作用も実装されている。代表的なライブラリと配布元を以下に示す。

【連立一次方程式ソルバ】

- PETSc+Kokkos (https://petsc.org/release/) :
- Ginkgo (https://github.com/ginkgo-project/ginkgo)

【固有値ソルバ】

- SLEPc (https://slepc.upv.es/): PETSc の固有値プラグインを提供する
- PHIST (https://bitbucket.org/essex/phist/wiki/Home)
- (P)FEAST (http://www.feast-solver.org/)
- 【連立一次方程式・固有値ソルバいずれも含む統合パッケージ】
- Trilinos (https://trilinos.github.io/)

【前処理行列作用】

- Hypre (https://github.com/hypre-space/hypre) : 前処理機能を提供する

3.2.5.3.2.4 Sparse direct solvers and hierarchical matrix solvers(疎行列向け直接法ソルバ・階層的行列ソルバ)

疎行列をLU分解などの直接解法により求解するライブラリである。 一般に直接法は反復法に比べ悪条件問題を解く能力が高い。計算複雑さは反復法に比べて高く、メモリを消費するなど巨大問題に適用することは容易ではない。 問題の特性と計算機の能力のトレードオフで選択する。

- SuperLU、SuperLU_MT、SUPERLU_DIST (https://portal.nersc.gov/project/sparse/superlu/)
- MKL Pardiso : MKL に含まれる共有、分散並列環境での直接法ソルバ
- MUMPS (https://mumps-solver.org/)
- STRUMPACK (https://github.com/pghysels/STRUMPACK)
- ButterflyPACK (https://github.com/liuyangzhuan/ButterflyPACK)
- Dissection(https://github.com/FreeFem/FreeFem-sources/tree/master/3rdparty/dissection、FreeFEM++から取得可能)

また、先節Trilinosの統合パッケージ内にもAmesos、Plirisなどの直説法ソルバが含まれる。なお、階層的行列ソルバについては調査中である。

3.2.5.3.2.5 その他（中間調査報告書作成において未調査または要調査のもの）

FFT(高速フーリエ変換)

継続調査中：

- heFFTe: Highly Efficient FFT for Exascale、ECP で開発された FFT (https://icl.utk.edu/files/publications/2020/icl-utk-1462-2020.pdf, https://github.com/af-ayala/heffte)。
- SPIRAL: CMU で開発されている FFT。SPIRAL 自身はカーネルコード生成機能だが生成物としての FFT カーネル群をここでは意味する(https://spiral-software.github.io/spiral-software/examples/basic/firstfft.html)。
- FFTX: ECP で開発された FFT。FFTW を代替する目的で API が統一化されており、内部的には SPIRAL と heFFTe の系統を統合した形の構成となっている (https://commons.lbl.gov/display/FFTX/FFTX+Home)。
- FFTW: MIT で開発されてきた FFT である。開発やメンテナンスが終了しており、多くはボランティアによる保守にとどまっている。対応する FFT 機能やプラットフォーム間のポータビリティの高さから多くの利用者が存在する (https://www.fftw.org/)。
- FFTE: 筑波大高橋大介教授による FFT である。近年は SPIRAL で生成された ArmSVE 向けカーネルを取り込むなどのバージョンも公開している (http://www.ffte.jp/)。
- MKL FFT：Intel MKL で提供される FFT (https://www.intel.com/content/www/us/en/develop/documentation/onemkl-developer-reference-c/top/fourier-transform-functions/fft-functions.html)。
- cuFFT: CUDA 環境向けの FFT (https://docs.NVIDIA.com/cuda/cufft/index.html)。
- rocFFT: ROCm 環境向けの FFT (https://github.com/ROCmSoftwarePlatform/rocFFT)。

Frameworks (フレームワーク)

調査中：Trilinos、PETSc、MFEM、SUNDIALS、deal II、ExaGO、HiOP

Mesh (格子生成)

調査中：Omega_h、PUMI、tetgen、gmsh、mmg3d、Trilinos

Graph (グラフデータライブラリ)

調査中：METIS、SCOTCH

DSL(Domain Specific Language)

プログラム環境において調査がなされており、当該調査を参照としたい

Tensors、convolution、decomposition、distributed format(テンソル・畳み込み演算・分散フォーマット関係)

調査中：Cyclops Tensor Framework、HPTT、ROTE、ExaTENSOR、Genten、Grace、PASTA

Pseudo Random Number Generator (疑似乱数生成器)

調査中: MTメルセンヌツイスターなど

3.2.5.3.3　数値ライブラリの機能別調査

序でもまとめたように数値計算ライブラリは、高度な数学とプログラミング技術が投入され作成されている。多くの利用者が信頼して利用するために、高性能であることはもちろん、その他の評価項目においてもその優劣は高いものを選択したい。 数値計算ライブラリの利用において顕著な理由は正確に問題を解くことにある。近代のプロセッサ技術の進展において低消費電力、それを実現するための低精度演算器の導入、さらにはそれらを個別パッケージ化したアクセラレータボードもしくは特殊CPU利用に関するプログラミングや実行環境との連携が重要である。本節では、まず既存の数値ライブラリが対応する計算精度に対して比較と分析を行う。引き続き、プログラミング言語対応を調査し実行環境などの継続調査項目へと引き継ぐこととする。

なお、中間報告時点では最新バージョンに対しての調査を行っているが、バージョンの更新によって結果が異なるため、今後の調査報告書の更新時において正確なバージョン表記を行うことを心掛ける予定である。

3.2.5.3.3.1 数値ライブラリの対応精度調査

各数値計算ライブラリが対応する浮動小数点フォーマットをまとめた。整数型等の浮動小数点以外のフォーマットにも対応するものがあればその他欄に記載した。横断的に、主要な数値アルゴリズムから代表的なライブラリを選択し比較している。従来から標準的な浮動小数点数であるFP64(倍精度)にはほとんどのライブラリが対応している。GPUの定着以後はFP32(単精度)の対応も進んでいる。さらにFP16（半精度）やさらに低精度への対応も試みられている。また、整数や低精度浮動小数点数を組み合わせたBLASなども実装されている。 AI分野における畳み込み演算に利用されていると分析される。

なお、富士通SSL IIが提供するBLASではFP16に一部対応している。詳細は付録で説明する。

表 3.2.5

		FP16	**BF16**	**FP32**	**FP64**	**高精度**	**混合精度**	**その他**
BLAS	**Netlib 版 BLAS**	×	×	○	○	×	×	
	Netlib XBLAS	×	×	○	○	80-bit, DD	○	
	OpenBLAS	×	△	○	○	80 bit	△	△INT8
	MKL 版 BLAS	△	△	○	○	×	△	△INT8, △INT16, △INT32
	SSL II 版 BLAS	△	×	○	○	×	×	
	SSL II 版 XBLAS	×	×	○	○	DD	○	
	cuBLAS	△	△	○	○	×	△	△INT8, △TF32
	rocBLAS	△	△	○	○	×	△	△INT8
	BLIS	×	×	○	○	×	×	

	cuBLASLt	○	○	○	○	?	?	△FP8, △ INT8, △ TF32
	cuBLASXt	×	×	○	○	×	×	
	cuSPARSE	△	△	○	○	×	△	△TF32
LAPACK	**Netlib LAPACK**	×	×	○	○	×	△	
	MKL LAPACK	?	?	○	○	?	△	
	cuSolverDN	○	○	○	○	×	?	△TF32
	rocSOLVER (a work-in-progress implementation of a subset of LAPACK functionality)	×	×	○	○	×	?	
BLACS/PBLAS/ ScaLAPACK	**Netlib BLACS/PBLAS/ ScaLAPACK**	×	×	○	○	×	?	
	MKL BLACS/PBLAS/ ScaLAPACK	×	×	○	○	?	?	
FFT	**FFTW**	×	×	○	○	△	?	
	MKL FFT	?	?	○	○	?	?	
	cuFFT	○	○	○	○	?	?	
	rocFFT	×	×	○	○	?	?	
	FFTX	?	?	?	?	?	?	
Random number generator	**MKL VS**							
	cuRAND	×	×	○	○	×	×	
	rocRAND	○	×	○	○	×	×	unit8/16/32/64

○　ある

△　一部ある

×　ない

？　未調査、暫定的結果（調査不十分）

3.2.5.3.3.2 プログラミング言語別利用可能調査

各数値計算ライブラリがどのプログラミング言語から呼び出し可能かを調査し、まとめた。中間報告の現段階では調査対象はBLASならびに固有値ソルバに限定される。

近年盛んに利用されているPythonについてはネイティブな実装系はほとんど想定されないため、バイナリオブジェクトやシェアードライブラリを呼び出す形を想定するため、数値ライブラリ開発者による公式な言語対応が明確でないものについて「？」をマークした。原理的には、Fortranで書かれたライブラリであっても

C/C++のラッパーを介して利用は見込めるため、今後最終調査報告書のための調査においてコミュニティ等による開発や公開が明確となれば「△」マークなどの修正を行いたい。

BLASやLAPACKなど伝統的な数値線形計算ライブラリはFortranが参照実装となっているため、Fortranからの呼び出しが標準であり、CやC++からも呼び出せるようなラッパ(cblasなど)を介して各種言語から利用が可能。GPU環境ではもともとC/C++が標準であるため対応言語がC/C++となっているものが多い。固有値計算ライブラリはScaLAPACKなどとの機能を一部利用するためFortran実装が多いと考えられる。下位層のツール群が十分に整備されればC、C++ネイティブの最適実装が出現する可能性は高い。

なお、extended-BLASの項はBatched形式の呼び出し方式や特殊なデータ形式・混合精度計算（本来は前節に含まれる分類ではあるが）を含めている。

表 3.2.6

		version	Fortran	C	C++	Python
BLAS	**OpenBLAS**	0.3.22	◎	◎	○	△
	MKL BLAS	2023.1	◎	◎	○	△
	cuBLAS	12.1.0	○	◎	◎	△？
	rocBLAS	2.47.0	◎	◎	◎	？
	BLIS	0.9.0	△？	◎	◎	？
extended-BLAS	**MKL batched**	2023.1	◎	◎	○	？
	MKL gemmt/matcopy	2023.1	◎	◎	○	？
	MKL gemm_bias (DPC++)	2023.1	×	×	◎	？
	cuBLAS batched	12.1.0	○	◎	◎	？
	cublas gemmEX (mixed precision)	12.1.0	○	◎	◎	？
	cuBLASLt	12.1.0	×？	◎	◎	？
	cuBLASXt	12.1.0	○	◎	◎	？
	rocBLAS batched	2.47.0	？	◎	◎	？

Sparse direct solver	**UMFPACK**	4.4.4	◎	◎	○	？
	SuperLU	5.3.0	◎	◎	○	？
	MUMPS	5.5.1	◎	◎	○	？
	Dissection	1.4.0	◎	○	◎	？
	MKL Pardiso	2023.1	◎	◎	○	？
Sparse iterative solver	**IML++**	1.4a	×	×	◎	×
	Hypre	2.27.0	○	◎	○	◎
	GAMG	3.19	◎	◎	○	？
	PETSc KSP solver	3.19	◎	◎	○	？
	MKL RCI CG/GMRES	2023.1	◎	◎	○	？

Dense eigenvalue solver	**ELPA**	2022.11.001	◎	◎	?	◎
	EigenExa	2.12	◎	?	?	△
Sparse eigenvalue solver	**ARPACK**	2.1	◎	△	△	× ※
	ARPACK-NG	3.8.0	◎	◎	○	○
	FEAST	4.0	◎	◎	○	?
	CISS (※SLEPc のモジュール)	3.19.0	◎	◎	○	?
	z-PARES	0.9.6	◎	?	?	?

◎ 可

○ 主ライブラリ関係団体が提供する wrapper 等を経由して呼び出し可

△ 主ライブラリ関係団体以外が提供する wrapper 等を経由して呼び出し可

× 不可（呼び出し手段、未発見）

？ 未調査、暫定的結果（調査不十分）

3.2.5.4 来年度計画

本中間報告時点では、数値ライブラリの動向調査において、主に密行列から疎行列の数値線形ライブラリの洗い出しといくつかの評価軸(計算精度、プログラミング言語の利用相関)に基づく特性をまとめた。2023年度以降も継続して調査を進め、数値計算ライブラリの完全調査を完了したい。

調査研究の主たる目的である次世代計算機につながる以下の項目についても、初年度の調査結果を発展させて進めていく。

1) 数値アルゴリズム自体を将来にわたって継続的に維持するために必要なライブラリ群ならびに不足するパーツについても調査を進める。 最終的には国内外で共同開発をすべきかなどの指針につなげることを目指す。これらの調査には、ベンダや計算機センタへのヒアリングにとどまらずアプリケーション・アルゴリズム調査グループとの議論を通じて進めていくことが肝要である。
2) 評価軸として設定した「計算精度」、「プログラミング言語の相関」についても精査を続けるとともに、新たに「消費電力」や「他ライブラリ間の相関」を調査し、次期システムが想定するモデルとの相関や数値ライブラリの予備的な性能予測へと展開したい。
3) 現状において性能的な問題点を有するライブラリも存在するが、次期システムに向けて性能面での問題点を洗い出すことにも注力し、数値計算ライブラリ構成方式の調査ならびに次期システムへの適合可能性につながる準備調査についても開始し、本格調査委へとつなげたい。

付録：富岳における富士通ライブラリの計算精度への対応状況

精度の対応状況

数値表現の半精度と四倍精度への富岳における対応状況を記載する。数学ライブラリと共に関係の深いコンパイラについて対応状況を記載する。

半精度への対応状況

binary16（IEEE IEEE 754-2008-2008 半精度浮動小数点型）に対して、富士通製の数値計算ライブラリでは表 3.2.7で示すBLASの一部のルーチンを提供している。なお、機械学習向け16bit表現であるbfloat16は未サポートである。

表 3.2.7

	Fortran BLAS ルーチン名	CBLAS ルーチン名	機能	並列 版
Level 1	FJBLAS_SWAP_R16	fjcblas_swap_r16	$x \Leftrightarrow y$	○
	FJBLAS_SCAL_R16	fjcblas_scal_r16	$x \leftarrow ax$	○
	FJBLAS_COPY_R16	fjcblas_copy_r16	$y \leftarrow x$	○
	FJBLAS_AXPY_R16	fjcblas_axpy_r16	$y \leftarrow ax+y$	○
	FJBLAS_DOT_R16	fjcblas_dot_r16	$\mathrm{dot} \leftarrow x^{\mathrm{T}}y$	○
	FJBLAS_ASUM_R16	fjcblas_asum_r16	$\mathrm{asum} \leftarrow \|\mathrm{re}(x)\|_1+\|\mathrm{im}(x)\|_1$	○
	FJBLAS_AMAX_I32_R16	fjcblas_amax_i32_r16	$\mathrm{amax} \leftarrow \mathrm{argmax}\ x_k$	○
Level 2	FJBLAS_GEMV_R16	fjcblas_gemv_r16	$y \leftarrow aAx+by$	○
	FJBLAS_GER_R16	fjcblas_ger_r16	$A \leftarrow axy^{\mathrm{T}}+A$	○
Level 3	FJBLAS_GEMM_R16	fjcblas_gemm_r16	$C \leftarrow aAB+bC$	○

コンパイラについてのbinary16への対応状況を表 3.2.8に記す。

表 3.2.8

言語種別	コンパイラ	型表現	データ表現	基本演算の実装	型変換	数学関数・基本入出力
C 言語	富士通 Trad	×	×	×	×	×
	富士通 Clang/LLVM7	_Float16	binary16	◎	◎	×
		__fp16	binary16	○	◎	×
	Clang/LLVM16	_Float16	binary16	◎	◎	×
		__fp16	binary16	○	◎	×
	gcc(12.2)	_Float16	binary16	◎	◎	×
		__fp16	binary16	○	◎	×
C++	富士通 Trad	×	×	×	×	×
	富士通 Clang/LLVM7	_Float16	binary16	◎	◎	×
		__fp16	binary16	○	◎	×
	Clang/LLVM16	_Float16	binary16	◎	◎	×
		__fp16	binary16	○	◎	×
	g++(12.2)	__fp16	binary16	○	◎	×
Fortran	富士通	real(kind=2)	binary16	◎	◎	△
	Flang/LLVM16	real(kind=2)	binary16	◎	◎	×(総称名の場合は単精度浮動小数点型へ拡張)
		real(kind=2)	binary16	○	◎	×(総称名の場合は単精度浮動小数点型へ拡張)
	gfortran(12.2)	×	×	×	×	×

・◎：native命令で実装、○：IEEE IEEE 754-2008 単精度浮動小数点型へ拡張、△：一部サポート、×：未サポート

bfloat16へは、富士通コンパイラ、Clang/LLVM16、GCCとも、基本演算・型変換・数学関数・基本入出力は未サポートであるものの、Clang/LLVM16に関してはC/C++でbfloat16型のACLE関数呼び出しの翻訳は可能である。

四倍精度への対応状況

富士通製の数値計算ライブラリはbinary128（IEEE IEEE 754-2008 四倍精度浮動小数点型）には対応していないものの、代入・四則演算や数学関数などの基本演算機能をdouble-double（IEEE IEEE 754-2008 倍精度浮動小数点型の変数2つで表現する四倍精度浮動小数点型）形式のライブラリとして提供している。

コンパイラについてのbinary128とdouble-double への対応状況を表 3.2.9に記す。

表 3.2.9

言語種別	コンパイラ	型表現	データ表現	基本の演算・入出力の実装	数学関数
C 言語	富士通 Trad	long double	binary128	libgcc (glibc)	×
	富士通 Clang/LLVM7	long double	binary128	libgcc (glibc)	libm
	Clang/LLVM16	long double _Float128	binary128	libgcc (glibc)	libm
	gcc(12.2)	long double _Float128	binary128	libgcc (glibc)	libm
C++	富士通 Trad	long double	binary128	libgcc (glibc)	×
		dd_real	double-double	富士通製ライブラリ (fast_dd)	富士通製ライブラリ (fast_dd)
	富士通 Clang/LLVM7	long double	binary128	libgcc (glibc)	libm
	Clang/LLVM16	long double	binary128	libgcc (glibc)	libm
	g++(12.2)	long double	binary128	libgcc (glibc)	libm
Fortran	富士通	real(kind=16)	binary128	libgcc (glibc)	富士通製ライブラリ (libfj*)
		dd_real	double-double	富士通製ライブラリ (fast_dd)	富士通製ライブラリ (fast_dd)
	Flang/LLVM16	real(kind=16)	binary128	libgcc (glibc), Flang ランタイム	libm, Flang ランタイム
	gfortran(12.2)	real(kind=16)	binary128	libgcc (glibc), gfortran ランタイム	libm, gfortran ランタイム

・×：未サポート

3.2.6 AI フレームワーク

3.2.6.1 調査目的

AI手法は現代科学において重要かつ不可欠なものとなり、スーパーコンピューティングシステムにおける主要なワークロードの1つとなった。FSプロジェクトのこの部分の目的は、人工知能ワークロードを実行するための次期フラッグシップスーパーコンピュータの準備状況を評価し、促進することである。

3.2.6.2 調査内容

本編の内容は、以下の通りである：

- ソフトウェア・アーキテクチャの観点から深層学習フレームワークの設計を調査する。
- 新しいハードウェアターゲットに対するサポートと最適化を追加できる最適な方法を特定する。

- 深層学習ワークロードを高速化するために、最近どのような種類のハードウェア機能が開発されているかを調査し、対応するソフトウェアサポートがどのように追加され、深層学習ソフトウェアスタックに統合されるかを調査する。
- ニューラルネットの計算複雑性の主要部分がどのカーネルに集中しているのか、また将来的にどのような傾向が予測されるのかを明らかにする。

3.2.6.3　調査結果

我々はニューラルネットワークアーキテクチャの動向を調査し、Transformerベースのモデルが最も顕著なワークロードの1つであることを確認した。Transformerモデルにおける主要な計算カーネルは、バッチ式行列乗算（BMM）である。行列の乗算は、より伝統的なタイプの層やAttention層のテンソル次元で使用されてきたが、BMMは、畳み込み層のフィードフォワードなどで計算が行われる方法とは異なるものにしている。

Transformer層では、マトリックスサイズは通常よりはるかに小さく、バッチ次元は大きくなる。これは、BMMのバッチ次元が、トレーニングデータのバッチ次元と、モデルで使用されるAttention headの数の積であるために起こることである。このため、A64FXのようなHPCアプリケーションに特化したプラットフォームであっても、優れた効率を達成することは非常に困難である。

我々は、様々な一般的なTransformerベースのモデルをプロファイルし、それらで使用されるバッチ行列乗算のカーネルサイズを抽出した。これらのサイズを、私たちが開発している深層学習ベンチマークフレームワークの新バージョンに追加した。私たちは、（BMMで使用される）小さな行列をA64FXなどの現在のアーキテクチャに最適化するためのいくつかのアプローチを開発し、いくつかの方法が将来のシステムに移行できることを期待している[図 3.2.16]。

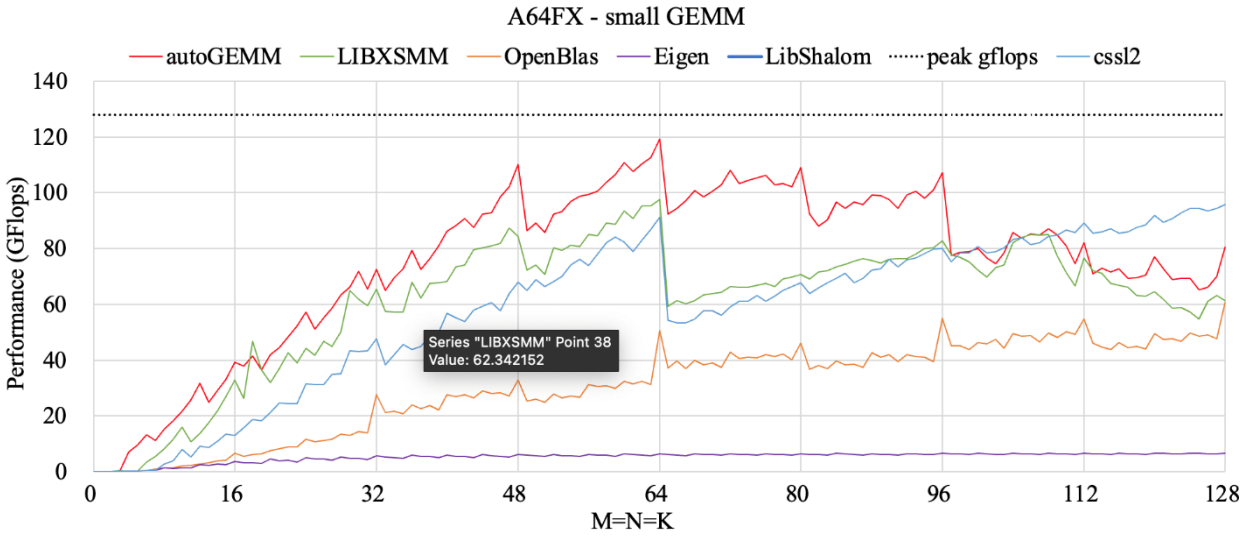


図 3.2.16 小さな行列のための最適化された行列乗算（GEMM）

混合精度と専用の行列乗算器の影響を調べたところ（図 3.2.17）、Transformer層は、フィードフォワードや畳み込みなどの層と比較して、混合精度をより高度に利用できることがわかった。しかし、LSTMベースのモデルや2D畳み込みネットワークは、TF32精度で行列乗算器を使用した場合でも約2.5倍のスピードアップを実現している。しかし、3次元畳み込みネットワークは、非常に小さなスピードアップを示していた。これは、3次元畳み込みが行列乗算によらず、「直接」、すなわちステンシル的な実装によって計算されることに起因している。

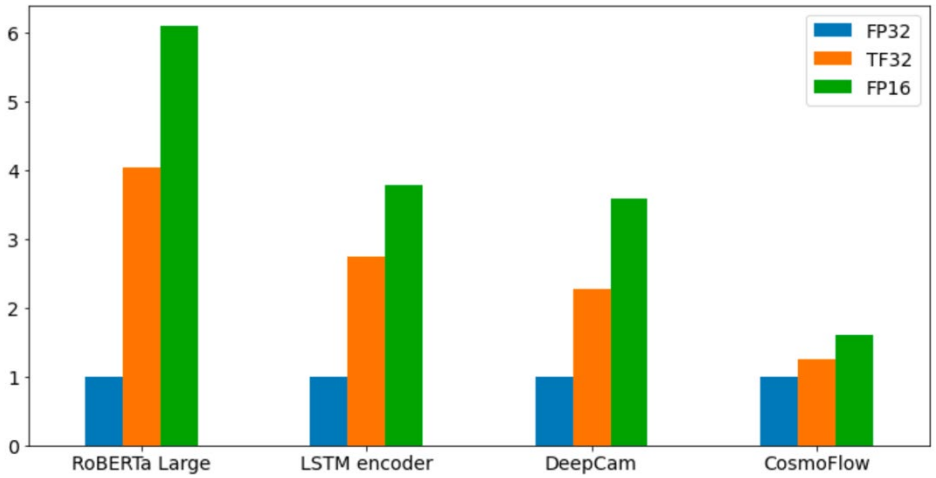


図 3.2.17 FP32 から TF32、FP16 の数値フォーマットに切り替えた場合の深層学習モデルの正規化された高速化

我々は、ソフトウェア・アーキテクチャの観点から、深層学習フレームワークの分析を行った。主なケーススタディとして、科学者が最も多く使用しているPyTorch深層学習フレームワークを選択した。私たちは、新しいハードウェアのサポートを追加できる抽象化レイヤを特定した。

その結果、深層学習に特化したパフォーマンスプリミティブのライブラリや、BLASなどの汎用的な計算カーネル（図 3.2.18の点線部分）が、最適化の対象として最もインパクトがあることが分かった。ただし、特定のカーネルの非同期呼び出しを容易にするなど、ソフトウェアの上位階層で行うべき作業もある。

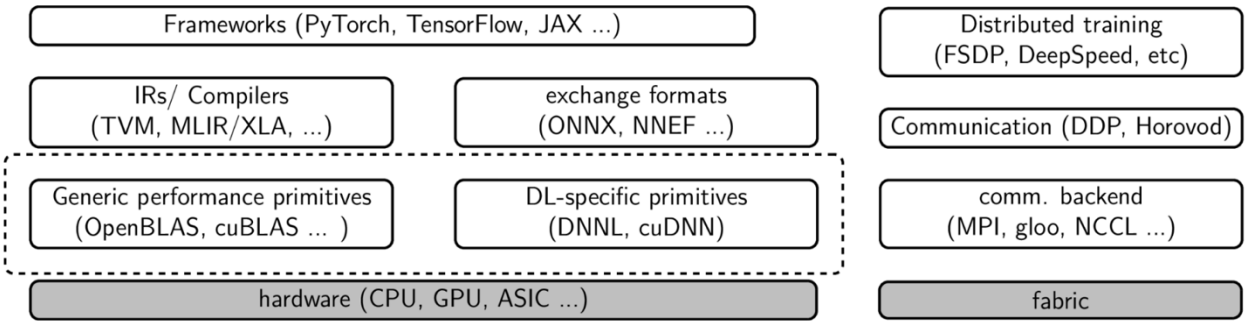


図 3.2.18 深層学習ソフトウェアにおける抽象化層の階層を模式的に示す図

また、新しいプロセッサをサポートするための有望な候補を検討した： NVIDIAのcuDNN、TVM、ARMのNN、IntelのoneDNNなどである。また、深層学習フレームワークの多くが持つ、特にCPUプラットフォーム向けの「ネイティブ」実装に注目し、そこで最適化されたコードを提供するという方法もある。この方法の問題点は、当然ながら、異なる深層学習フレームワークへの移植性に限界があることである。

NVIDIAのcuDNNは、最適化された深層学習プリミティブの最も有名なライブラリの1つであろう。このライブラリのソースコードは公開されておらず、アーキテクチャのターゲットもNVIDIA GPUに限定されている。

しかし、パイロットスタディでは、GPUデバイスのモックを作成し、いくつかの選択されたカーネル（例えば、ストラ

イドとダイレーションのない一般的なサイズの畳み込み）の対応するAPIコールを、単純なBLASベースの実装[Moses et al. PPoPP'23] にリダイレクトすることに成功した。このアプローチの利点は、深層学習フレームワークはGPUターゲットに大きく最適化される傾向があり、これにはプリミティブライブラリ以外のコード、例えばカーネルの非同期ディスパッチのサポートが含まれることである。

TVMとARMのNNは、ARM CPUをメインターゲットまたは可能なターゲットの1つとして明示的にサポートするという利点がある。しかし、どちらのソリューションも推論に重点を置いており、トレーニングのサポートは初歩的なものであった。

最後に、A64FXプラットフォームへの深層学習フレームワークの性能移植を可能にするベースとして、インテルのoneDNNが選ばれた。以前はMKL DNNとして知られていたこのライブラリは、もともとIntel x86 CPUでの推論をターゲットとして設計されていたが、最終的には、推論だけでなくトレーニングにも対応した、より幅広いターゲットを持つオープンソースプロジェクトに発展した。DNNLにA64FXのサポートを追加するには、まずXbyak JITアセンブラにA64FXターゲットを追加し、次に個々のカーネル用にカスタム実装を書く必要があった。oneDNNの富岳での経験は、oneDNNを新しいプロセッサに対応するためのベースとして利用することの可能性を示しているが、それでも高いエンジニアリングコストが必要である。

3.2.6.4　来年度計画

まず、どのようなアーキテクチャの特徴がTransformerモデルに利益をもたらすかを調査する予定である。 具体的には、与えられた演算性能でどれだけのメモリ帯域幅が必要なのか、行列乗算器や超広帯域ベクトルを使用することでどれだけの加速が可能なのかなどである。

大規模なLLMのトレーニングが実証されている[Narayanan SC'21]。特に、NVIDIAのMegatron-LMは、エンドツーエンドのトレーニング（通信、データ処理、最適化を含む）において、3,072個のA100 GPUで1Tのパラメータを持つモデルの理論ピークFLOPSの～50%を達成することができる。このレベルのスケーリングには、並列性（データ並列、テンソル並列、パイプライン並列）を実現するための精巧なハイブリッドアプローチが必要である。現在のシステムソフトウェアスタック（NVIDIA GPUシステム用）は、このようなスケールを実現することが可能である。Megatronのようにシステムソフトウェアスタックを～50%の効率にできるレベルに持っていくための富岳での継続的な努力は、新しいシステムでLLMをスケールさせるために大規模なエンジニアリング努力が必要であることを示す。特に、a)シングルノードレベルのプリミティブの最適化、b)大きなメッセージサイズに対する集団通信のスケーリング、c)ハイブリッド並列のサポートが挙げられる。

今後、LLMをスケールアップさせる方法として、これまで考慮されてこなかった課題もある。まず、LLMで大きな入力（シーケンス）長を使用することに関心が高まっている。これは、出力の質を向上させることができるからである。LLMの注意層の計算量とメモリは、配列長に対して二次関数的に増加するため、LLMのスケーリングは困難であることが予想される。我々は、シーケンスを並列化する方向で調査した。このアプローチをサポートするためには、スパース法の改良も必要かもしれない。第二に、モデルの品質は利用可能なデータに依存するため、LLMで使用されるデータは、モデルサイズの増加よりも高い割合で増加している。例えば、GPT4モデルのパラメータ数はGPT4の6倍であるのに対し、GPT4の訓練に使用されるデータはGPT3の20倍である。データセットの調達、クリーンアップ、準備のためのソフトウェアパイプラインは、規模が大きくなるとボトルネックになり始める可能性がある。第三に、マルチモーダルLLMは、異なるタイプのデータやタスクを扱うことができるため、広く関心を集めている。このことは、現在のところまだ十分に理解されていないモデル（例：動的挙動）をスケールアップする際の課題をもたらす可能性がある。

リカレントネットワークのような既存のアーキテクチャを見直す試み、スパースモデル、局所更新ルール、神経科学から着想を得た新しいパラダイムなど、新しいアプローチも検討している。

AI、特にディープニューラルネットワークは、劇的なまでに規模を拡大し、複雑化している。例えば、生産レベルのトランスフォーマーは、わずか5年で100万倍もの成長を遂げた： 2017年のBERT、2022年のGPT3。ハイパーパラメータのオートチューニングは、モデルのスケーリング時にパフォーマンスのために必要になってくる。OptunaやTVMなどのオートチューニングツールは、ディープニューラルネットワークで広く使用されている。しかし、モデルの進化速度が速いため、AIを対象としたオートチューニングツールに求められる要件を定義する必要がある。次のフェーズでは、その要件を明らかにすることを一つの目標としている。

3.2.7 OS・仮想化・クラウド連携

3.2.7.1 調査目的

次世代情報基盤のためのオペレーティングシステム(OS)および仮想化技術に関する調査、検討を行う。次世代情報基盤では、メニーコアCPUやアクセラレータの導入が加速し、コンテナのような仮想化技術の必要性は高まっている。性能、省電力、ユーザビリティ、セキュリティの観点で、次世代スーパーコンピュータ向けのOSおよび仮想化技術について調査検討する。

3.2.7.2 調査内容

2022年度は、既存スパコンにおけるOS利用状況と軽量OSに関する調査、HPC分野における仮想化技術の動向調査、およびアプリケーションユーザからのOS、仮想化技術、クラウド連携への要望調査を行った。クラウド連携については、運用技術調査研究において調査を進めており、本報告書では割愛する。

3.2.7.3 調査結果

本年度は、(1) 既存スパコンにおけるOS利用状況と軽量OSに関する調査、(2)HPC分野における仮想化技術の動向調査、および(3)アプリケーションユーザからのOS、仮想化技術、クラウド連携への要望調査を行った。

(1) 既存スパコンにおけるOS利用状況と軽量OSに関する調査

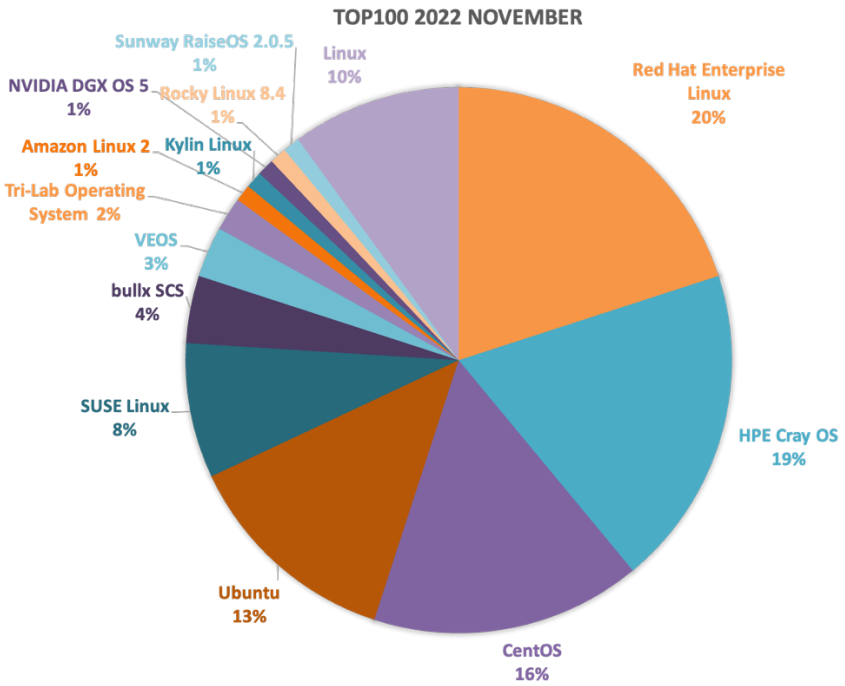


図 3.2.19　TOP500 の上位 100 スパコンの OS 内訳

既存スパコンのOS利用状況調査として、2023年11月のTOP500リスト[1]の上位100位のスパコンのOSの内訳を図 3.2.19にまとめた。図 3.2.19から、 TOP100スパコンではすべてLinux系OSが採用されており、Red Hat Enterprise Linux、HPE Cray OS、SUSE Linux等の商用OSが半数を占めている。CentOS、Ubuntuなどのオープンソースのディストリビューションの利用も多いことがわかる。一方、わずかではあるが米国LLNLが開発しているTOSS (Tri-Lab Operating System Stack)や中国のSunway Raise OSなど、Linuxをベースとして独自に開発されたOSも利用されている。よって、今後もLinuxをベースとしたOSの利用が続くと考えられる。なお、軽量カーネルの利用の有無については記載がなかった。

OSノイズを軽減して並列性能向上を図るため、軽量OSの開発も進められている。代表的なものとして、ANLのArgo [2]、SandiaのHobbes [3]、IntelのmOS [4]、理研のMcKernel [5]がある。McKernelはOSSとして公開されており、現在も東大で研究が継続されている。理研の報告[6]では、10万台規模の並列処理でOSノイズにより60%の遅延が発生するのに対し、アシスタントコアの利用で0.1%、McKernelの利用で0.009%に遅延が削減できることが示されている。また、多くの並列アプリケーションプログラムでも軽量カーネルの利用による性能向上が示されている[7]。一方、スパコン向けOSにおいてもOSノイズを削減する試みが進められており、軽量OSとの優位な性能差がなくなりつつある。

(2) HPC分野における仮想化技術の動向調査

仮想化技術の動向調査としては、SC22の併設ワークショップであるCANOPIE-HPC (International Workshop on Containers and New Orchestration Paradigms for Isolated Environments in HPC)[8]での発表について、表 3.2.10にまとめた。

表 3.2.10 CANOPIE-HPC の発表の概要

	概要	コンテナの利用目的／高度化
招待講演[9]	LHC ジョブ実行基盤に K8s 採用	再現性、インフラ管理と配備の簡素化、スケーラビリティ
[10]	ハイブリッドクラウドでマルチスケールサイエンスワークフロー計算基盤構築	シミュレーションモジュールのコンテナ化、ポータビリティ
[11]	実験結果の詳細な解析、再現性、ポータビリティのため、Provenance Capture System を開発	再現性、ポータビリティ
[12]	Podman を HPC 用コンテナに採用。ルートレスモードでセキュリティの課題解決。遜色ない性能。	（Shifter からの移行）標準コマンド I/F も利用可能
[13]	セキュリティや管理のため、アプリとランタイムのコンテナを分離。K8s と Podman で実行確認。	K8s での HPC アプリ実行
[14]	Workflow オーケストレータの紹介	Workflow モジュールとして利用
[15]	MPI ライブラリとネットワーク HW のミドルウェアである libfabric を開発。ネイティブに近い性能。	コンテナの通信性能向上
[16]	HPC ワークフロー向けに K8s スケジューラプラグインを開発	K8s での HPC ワークフロー実行

招待講演として、CERNのRicardo Rocha氏より”Kubernetes and Cloud Native at CERN”という講演があった。CERNのLHCワークロード処理において、再現性やインフラ管理および配備の簡素化、スケーラビリティのためにCloud Nativeな技術であるKubernetes (K8s)を採用したという報告があった。また、K8sで静的分割、MPI、ML、AIのようなHPCワークロードに対応させるため、K8s Batch WGの活動を行うなど、HPC用途でK8sを活用する活動を行っている。

その他の発表においても、ライブラリのモジュール化やポータビリティ、再現性のため、コンテナを活用した研究開発が進められていることがわかる[10][11][14][16]。また、そのようなコンテナの利点をHPC環境で活用するため、コンテナを利用した実行環境の性能向上やユーザビリティの強化などを目的とした研究も複数発表されていた[12][13][15]。

(3) アプリケーションユーザからのOS、仮想化技術、クラウド連携への要望調査

- OS への要望調査

  アプリ開発者、利用者からのアンケートでは、次世代計算基盤のOSに対してUNIX系OS/Linuxの採用や現行OSとの連続性、安定性といった「使いやすさ」の要望が多かった。また、計算ノードの軽量化や最適化といった「性能」への期待に関する意見もあった。

- コンテナへの要望調査

  アプリ開発者、利用者からのアンケートでは、3割以上の研究者がコンテナを利用している、または利用を予定している、と回答しており、HPC分野における仮想化技術への期待は今後ますます高まると予想される。コンテナ利用については、ソフトウェアのパッケージ管理や既存コンテナイメージの利用、研究の再現性といった観点で利用されている。またコンテナを実行単位とした計算基盤の利用も進んでおり、DockerとSingularityが多く使われている。

HPCでのコンテナ利用について期待している点としては、上記の利用目的で挙がった点に加えて利用者によるイメージファイルの作成等が可能になることや、ホストライブラリとの整合性の課題の解決といった、簡単に使える環境の整備が求められている。

セキュリティの観点から、多くのスパコンではSingularityやShifterのようなルート権限をもたないコンテナ技術が採用されており、コンテナイメージの作成も一般利用者には制限されている。これにより不便を感じている利用者がいることが確認できた。一方、最近ではPodmanやDockerのルートレスモードなど、ルートレスで利用できるコンテナもある。セキュリティを維持しつつ利便性を高めていく必要があると考えられる。

- クラウド連携への要望調査

アプリ開発者、利用者からのアンケートでは、3割弱の研究者がすでにクラウドを利用しており、さくらのクラウド、AWS、mdx、Azure、Google Cloudを利用していると回答があった。クラウド利用の目的としては、データの管理、データ・計算資源の共有、アプリジョブ実行、プログラム開発、およびベンチマーキングが挙げられていた。次世代計算基盤のクラウド連携についての期待としては、大規模データの費用の問題の解決、（再翻訳なしで）シームレスに実行できる環境、シームスなデータ（アプリのデータ、コード、リポジトリを含む）のやり取り、webを用いた計算結果の可視化環境、クラウド的に使えるノードの提供、量子計算機やアニーラとの連携、などが挙げられた。よって、アプリケーションの移植性やデータ転送支援、ネットワークボトルネックの解消は重要な課題と考えられる。また、フラッグシップモデルシステムとクラウドでシームレスなジョブ実行、データ管理を実現するには、性能的な課題だけでなく、統一インターフェースの提供、アカウント管理、スケジューラの連携、ライブラリ管理等、様々な課題があり、運用技術調査研究チームにおいて調査を進めている。

3.2.7.4　来年度計画

2023年度は、HPC向けOSおよび仮想化技術について、より詳細な調査をすすめるとともに、アーキテクチャ調査研究グループおよびアプリケーション調査研究グループの調査結果をもとに、さらなる課題の抽出を行う。また、省電力、ユーザビリティ、セキュリティの観点での調査をすすめる。

[1] https://www.top500.org/

[2] https://web.cels.anl.gov/projects/argo/

[3] https://www.sandia.gov/ccr/project/hobbes-extreme-scale-operating-systems-project/

[4] https://github.com/intel/mOS

[5] https://github.com/RIKEN-SysSoft/mckernel

[6] https://www.fujitsu.com/jp/documents/about/resources/publications/technicalreview/2020-03/article06.pdf

[7] Balazs Gerofi, Kohei Tarumizu, Lei Zhang, Takayuki Okamoto, Masamichi Takagi, Shinji Sumimoto, and Yutaka Ishikawa. 2021. Linux vs. lightweight multi-kernels for high performance computing: experiences at pre-exascale. In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '21). Article 11, 1-13. https://doi.org/10.1145/3458817.3476162

[8] https://canopie-hpc.org/
[9] Invited Talk: Kubernetes and Cloud Native at CERN, Ricardo Rocha, CERN
[10] Multiscale Scientific Workflows on High-Performance Hybrid Cloud, Vadim Elisseev 他、IBM Research Europe 他, CANOPIE-HPC WS, 2022.
[11] Complete Provenance for Application Experiments with Containers and Hardware Interface Metadata, Quincy Wofford 他、LANL 他, CANOPIE-HPC WS, 2022.
[12] Scaling Podman on Perlmutter: Embracing a Community-Supported Container Ecosystem, Laurie Stephey 他, NERSC/LBNL 他, CANOPIE-HPC WS, 2022.
[13] A Separated Model for Running Rootless, Unprivileged PMIx-Enabled HPC Applications in Kubernetes, Joshua Hursey, IBM, CANOPIE-HPC WS, 2022.
[14] Invited Talk: Rethinking containers for cloud native pipelines, Ben Sherman, SeqeraLabs, CANOPIE-HPC WS, 2022.
[15] Libfabric-Based Injection Solutions for Portable Containerized MPI Applications, A, CANOPIE-HPC WS, 2022. Madonna & T, CANOPIE-HPC WS, 2022. Aliaga, Swiss National Supercomputing Centre, CANOPIE-HPC WS, 2022.
[16] One Step Closer to Converged Computing: Achieving Scalability with Cloud-Native HPC, Daniel J, CANOPIE-HPC WS, 2022. Milroy 他、LLNL 他, CANOPIE-HPC WS, 2022.

3.2.8 セキュリティ

3.2.8.1 調査目的

システムソフトウェア・ライブラリはHPCアプリケーションのユーザがソフトウェアを構築する際やプログラミングしていく上でインターフェースとなる。オープンソースの利用の普及や機械学習分野のように処理フレームワークの活発な開発により、システムソフトウェア・ライブラリは日進月歩のスピードで変化を続けている。また、アプリケーション分野によっては、スーパーコンピュータでおこなう処理やデータについてもセキュリティを高いレベルで確保する必要がある場合も散見される。このような状況に対して、システムソフトウェア・ライブラリは様々なソフトウェアスタックのレイアを重ねていくことにより要求するセキュリティを実現していくことが試みられている。従って、スーパーコンピュータにおけるセキュリティを考えていく上では、システムソフトウェア・ライブラリの単一のレイアの要素技術だけでは対応しきれない状況となりつつある。そこで、本調査研究では、システムソフトウェア・ライブラリの技術を俯瞰し、サブグループを横断してセキュリティの面での調査研究を実施する。システムソフトウェア・ライブラリに関する各サブグループと密接に連携し、各サブグループがまとめた技術動向調査に関してシステム全体からの視点での評価指標の選択や優先順位の検討を取りまとめていく。加えて、複数のサブグループをまたぐマルチレイアに渡る検討が必要なセキュリティの面での考察を、サブグループを横断する形で実施する。

3.2.8.2 調査内容

システムソフトウェア・ライブラリに関する要素技術についての調査研究の全体取りまとめを行うと同時に、サブグループを横断して検討する必要があるセキュリティ等の技術分野に関して先導的に調査検討を行う。要素技術にかかわる調査研究だけではなく、システム全体からの視点で検討することを試みる。そのために、アプリケーション調査グループからの理想とする将来のアプリケーション像・ソフトウェア像を調査し、その結果を踏まえて横断的な調査研究が必要なセキュリティの面での検討を進める。アプリケーション調査グループと連携して個々のユーザが必要とするセキュリティレベルを確保するための暗号化、改ざん防止技術（ブロックチェーン）等に関してのソフ

トウェア技術を調査し、演算性能やI/O性能等に与える因子を評価しながらソフトウェア技術全般に関わる横断的な評価と検討を進める。

3.2.8.3　調査結果

1つ目の取り組みとして、セキュリティの分野は商用クラウドの分野で非常に進んでいることから、商用クラウドで既に利用できるセキュリティに関するサービスやソフトウェアスタック、ソフトウェアの機能を調査するとともに、商用クラウドの分野の技術動向を調査研究した。商用クラウドを利用したセキュアなシステム構築においては、ゼロトラストと暗号化という2つの大きな流れがあるのでそれぞれについて調査した。

ゼロトラストとは、システムにログインできるアカウントを持っているユーザであってもシステムの脆弱性から認証に成功した可能性もあることなどから信用がないものとみなし設計するというコンセプトである。システムを設計する上でゼロトラストを前提とした考え方をする必要があるとされ、現状のスーパーコンピュータの利用の現場で行われているような公開鍵認証方式における鍵とパスフレーズによる認証がされたら基本的にすべての操作を可能とするのではなく、セキュリティ上の敵はシステム上のどこにいるかわからないという考え方の下で、権限の細分化とアクション毎の認証や、リソースや通信の監視を強化していくという方向性である。これらは、HPCアプリケーションの実行される環境に適応すると若干の性能的なオーバーヘッドがあることが予想される。そこで、性能低下の度合いを具体的に評価していく必要もあると考えられるため、次年度以降に継続して調査する計画である。また、システムが外部のセンサやクラウドのリソースと接続するケースも増えることが予想されることから、認証の方式などにおける課題も検討していくべきとの知見を得ている。

ゼロトラストの概念はソフトウェアに限ったものではないため、ハードウェアやシステムのゼロトラストという点での調査も行った。セキュリティ技術の面で先進的な取り組みを重ねているクラウドコンピューティングの分野では、クラウドのシステムの取得、開発および保守を行う際のセキュリティへの要求事項として、以下のようなものを挙げている[1]。

- 構成管理と、セキュアな開発に必要なものを考える必要もある
- サプライチェーン、事業者のリスクを考える必要もある

これらの必要事項をすべて自前で満たすことを確かめ、検証を行うことは非常に労力のかかることなので、現在、ISO/IEC 27017:2015として標準化され、ISMSクラウドセキュリティ認証として監査制度、認証制度を制定することにより外部機関にアウトソーシングしセキュリティの検証をしやすくする仕組みをクラウドコンピューティングの分野では取り入れている。これらは、一般的な情報システムのセキュリティ認証に対して、クラウド固有の追加として、利用における要求事項の分析と仕様化、開発プロセスの要求定義と確認することなどを加えている。これらに合わせて、クラウド利用者、クラウド提供者、運用・保守を行うパートナーなど、ステークホルダ間における責任関係の明確化と体制の整備が重要となるとクラウドの分野では言われている。これらの項目は、スーパーコンピュータに当てはめても仕組みとして十分成立すると思われる。今後、詳細な検討を行う予定である。

暗号化に関しては、鍵管理などのクリティカルな処理をユーザが使っているOSとは独立して安全に実行できる環境であるTEE（Trusted Execution Environment）が近年のCPUにはハードウェアの機能として含まれるようになってきている。例えば、Intel SGX, AMD SEV, ARM TrustZoneが有名である。TEEの環境を持つことにより、OSとは独立してセキュリティを担保するクリティカルな処理が安全に実行できるため、OSの脆弱性が突破されることによる秘密情報の漏洩や処理の改ざんのリスクをなくすことができる。TEEを実現するために、CPUは処理用の通常のCPUとは完全に独立した別チップやモジュールを用意し、鍵生成、機密情報保持、乱数生成などの機能をオフロードする。

既に商用のCPUに組み込まれているIntel SGX, AMD SEV, ARM TrustZoneについての概要を調査した結果を順に説明する。Intel SGXはEnclaveと呼ばれるメモリが暗号化され隔離された実行環境を動的に提供する。ユーザーレベルで隔離された実行環境を呼び出すことができるため、管理者ではなく使えるという利点がある。一方で、アプリケーション自体をEnclaveで動作させるためには、動作対象部分のプログラムの書き換えが必要となる。

AMD SEVはクラウド環境でVMをホスティングする際のセキュリティを高めることを主眼として設計されている。従って、クラウド環境の利用者が立ち上げたVMをクラウド環境の管理者から見ることができない鍵で暗号化することができ、VMの単位でTEE環境を構築することができる。ハードウェアを用いた一般的な暗号のエンコードとデコードの回路は入出力のバンド幅に合わせてラインスピードとなるようにデータの移動する速度に合わせてボトルネックなく処理するように設計することが可能である。そこで、AMD SEVはメモリコントローラに暗号のエンコードとデコードの回路を追加し、対象となるメモリを指定された鍵を用いてスループット低下をほぼない状態にて暗号化できる。AMD SEVはVMやハイパーバイザの単位で鍵を管理することにより、ユーザからは等価的にTEEを構築することができる反面、ユーザ権限でプログラムの一部をTEEで実行するという用途では使うことはできない。

ARM TrustZoneは最も広く使われているTEEであるが、用途がスマートフォンやゲーム機器、IoTデバイスなどコンシューマー向けの組込みシステムが中心であり、サーバでの利用の事例は現在のところ見られないようだ。ARM TrustZoneはSecure WorldとNormal Worldの2つの環境でOSを動かすことができ、ハイパーバイザ、管理者、ユーザといった異なる4つの特権レベルでの管理に対応できる。これにより、Secure Worldでは Trusted OSを動作させつつ、Normal WorldではLinuxのような汎用的なOSを動作させるということができる。一方で、Secure Worldで動作するTrusted OSはTEEを提供するがそのメモリの内容が暗号化されているわけではなく、Trusted OSの上で動作するアプリケーションにTEEの環境を提供することを目的としている。

暗号化技術としては、公開鍵認証方式だけではなく、近年、秘匿マルチパーティ計算に注目が集まっている。秘匿マルチパーティ計算は、秘匿計算、秘密計算、マルチパーティ計算と呼ばれることもあり、同じ意味で使われる。秘匿マルチパーティ計算は、それぞれ暗号化されたデータセットを持つ複数の参加者がそれぞれ自身の入力値を秘匿したままで多入力関数の値のみを計算することが可能となる暗号学的な手法である。一方で、HPCアプリケーションにおいて、直接的に秘匿マルチパーティ計算を計算ノードで処理する用途があまり想定できていない。例えば、複数の外部のクラウド上のサーバから暗号データを復号化することなく計算できるという用途はあるかもしれないが、ネットワーク帯域等の制限によりその計算頻度は限られたものになることが想定される。

2つ目の取り組みとして、運用とシステムソフトの切り分けの議論を行った。システムソフトウェア・ライブラリの様々なソフトウェアスタックやレイアにおけるセキュリティの技術分野は、スーパーコンピュータを実際に運用する上でのポリシー策定にも密接にかかわる。本グループにおけるセキュリティは、ユーザがソフトウェアやセキュリティの機能を選択できるかどうかの視点で考えるとして、ユーザが選択できないものは運用側の問題として取り扱わないこととした。ユーザが必要なセキュリティ保護に必要な機能や仕組みをスーパーコンピュータの環境で使えるかという面、あるいは、必要な機能や仕組みがオープンソースで提供されているとしてユーザ環境にインストールして使えるかという面、あるいは、管理者権限を持つことなくソフトウェアの構築や移植ができるかという権限の面での整理も必要であることが分かった。

3つ目の取り組みとして、アプリケーショングループにセキュリティに関するアンケートを実施し、将来的なスーパーコンピュータのアプリケーションに必要となるセキュリティに関するユーザからの要求を調査した。その結果、以下の2つの回答が寄せられた。

- docker を利用して環境構築出来ると嬉しい

- 現在、政策対応利用課題で利用している。複数機関の利用者が同一の課題に参加しているが、データの参照範囲について機関毎に制御したい側面もあり、同一課題で複数のグループ管理ができることを期待する。

1つ目の回答のDockerについては現状のスーパーコンピュータ富岳では利用できないが、よりHPC向けに仮想化による性能低下を抑えた実装であるSingularityが利用できる環境がスーパーコンピュータ富岳で提供されている。一方で、ユーザが立ち上げたコンテナでRootになることができないように設定されているため、Root権限を持つものが使う一般的なDockerの利用のイメージからは離れるものとなっていることによるユーザからのリクエストであるとも受け取れる。VMレベルで、Rootになれて、富岳でも動くハイパーバイザとしてMilvusVisor（ARM AArch64向け軽量ハイパーバイザ）が理研RCCSのGithubで公開されている[2]ので、今後、MilvusVisorなどを前提としたVM単位でのリソース提供の実現可能性なども検討するべきだということも課題として挙げられることが分かった。

2つ目の回答の権限の管理に関しては、ゼロトラストの観点からも、ユーザが必要だと思うのであればAction毎に細かな権限管理をする機能があると有用であり、今後、セキュリティ的にも重要となるであろう。

3.2.8.4 来年度計画

- ゼロトラストに必要な権限の細分化とアクション毎の認証や、リソースや通信の監視を強化していくための性能的なオーバーヘッドに関する調査
- VM 単位でのリソース提供の実現可能性の評価
- コンテナを用いた際のセキュリティ上の問題点の調査
- アプリケーション調査グループと連携したシステム横断的評価とセキュリティに関する検討の取りまとめ
- ユーザから透過的なメモリの暗号化による性能オーバーヘッドの評価
- HPC 分野で将来的に求められるセキュリティレベルについての調査

[1] 金子 格 編著、石黒正揮 、小川宏高、小向太郎、櫻田武嗣、千葉立寛、林 良一 著. クラウドシステム移行・導入―アーキテクチャからハイブリッドクラウドまで. オーム社（2022）
[2] https://github.com/RIKEN-RCCS/MilvusVisor

3.2.9 自動チューニング

3.2.9.1 調査目的

自動チューニング(Auto-tuning、AT)技術は、アルゴリズム、ソフトウェア、およびハードウェアに現れる性能パラメータを調整する技術であり、実行時や入力データに基づく性能向上の手段として有効な技術であることが知られている。本調査では、数値計算ライブラリ（アルゴリズム）を中心として、アプリケーションからシステム・ソフトウェアまでのAT適合性や有効性を調査し、次世代システムの性能見積もり、チューニング、利便性に資することを目的とする。

以下の2目的がある。

- 目的１：サブグループの縦串としての AT 技術調査
- 目的２：AT 最新技術調査

3.2.9.2　調査内容

- 目的１：サブグループの縦串としての AT 技術調査

[AT適用可能性の調査]：

アプリGのアプリケーション開発者と利用者に対するアンケートから、AT適用の要望調査を行う。

[一般からの意見交換]：

学会等においてパネル討論を実施することで、次世代システムに関する意見収集を行うとともに、AT機能に対する要求を調査する。

- 目的２：AT 最新技術調査

[ATソフト調査]：

米国エネルギー省（DOE）などで開発されているATソフトと機能を調査する。

[新奇ハードウェアのAT]：

新奇ハードウェアとして想定される量子コンピュータ（含む、関連技術）におけるAT適用可能性と効果を調査する。

3.2.9.3　調査結果

- 目的１：サブグループの縦串としての AT 技術調査

[AT適用可能性の調査]：

以下に、アプリGからのアンケートによる回答結果を示す。なお、以下は回答を、システム・ソフトウェア上のレベルで分類したものである。

[問] 扱っているプログラムにおいて、自動チューニングしたい対象（例：アルゴリズムパラメタ）があれば、教えてください。

表 3.2.11 アプリ G からのアンケート結果のまとめ

アプリケーション アルゴリズムレベル	ライブラリレベル （含、AI ツール）	システム・ソフトウェア レベル	ハードウェア レベル
・物理変数の精度 （FP16～32～64） ・パラメタリゼーション内のパラメータ ・コアの磁気流体計算部分 ・不確定な物理パラメータ雷や雲の成長過程の計算に含まれる様々なチューニングパラメタ ・ハイブリッドモンテカルロで用いる分子動力学のパラメータ ・データ同化 ・気象モデルの一つのコンポーネントである雲微物理モデルに多数含まれるチューニングパラメタ	・マルチグリッドソルバの格子レベルや ・並列分割粒度 ・ニューラルネットワークの学習時のメモリ量 ・線形ソルバの前処理 に関するパラメータ ・アルゴリズム中のパラメータ （ブロックサイズなど） ・計算領域の分割形状 ・ハイパーパラメータ	・計算と通信のオーバーラップ率、および 通信メッセージサイズ	・巨大疎行列-ベクトル演算でのキャッシュミス ・タイリングサイズ ・GPU スレッド数 ・GPU 対応

[一般からの意見交換]：

本年度は、以下の2つのパネル討論により議論を行った。1）AT研究会ワークショップ（ATTA2022、2022年12月23日開催）でのパネルディスカッション「"Next Fugaku"に向けた自動チューニング技術」；2）ATに関する日台ワークショップ（ATAT2023、2023年3月25日開催）「Next supercomputer system requirement from Japan and Taiwan」。

これらのパネルでは、以下の題目が議論された。

- 量子コンピュータの適用可能性
- 既存の数値計算ライブラリの仕組みで、次期システムでも性能向上ができるか？
- ベンチマークと、アプリ側からみた AT への期待ー生産性の向上ー
- 汎用コンパイラへの AT の埋め込み
- ワークフローへの対応ーヘテロジニアスアーキテクチャの観点からー
- 変動精度計算、精度保証への対応
- AI と HPC、およびそのジョブフロー対応
- デジタルツインと超並列処理
- アプリケーションの観点での精度・安定性・移植性
- 多様なジョブを取り扱うためのスケジューラ機能

● 目的２：AT 最新技術調査

[ATソフト調査]：

DOE関連開発のATソフト調査として、GPtune (https://github.com/gptune/GPTune) と、Ytopt （https://github.com/ytopt-team/ytopt)を対象とした。詳細調査は来年度に継続する

が、GPtuneについてはScaLAPACK QR、および、SuperLU_DISTに適用例がある。そのため、数値計算ライブラリ機能と共に調査研究を行う。

[新奇ハードウェアのAT]：

量子関連技術へのAT適用として、量子回路シミュレータ（NVIDIA cuQuantum）、疑似量子アニーリングマシンのハイパーパラメータの調査、および、AT効果の調査を行った[1]。AT効果については、限定問題であるが、疑似量子アニーリングでは同問題特有のQUBO形式によるエネルギー記述に現れるパラメータ調整をすることで、最適解回答率が0%から100%まで変化する報告がある。そのためチューニング効果は大きく、AT適用が必要とされる。一方、量子回路シミュレーションの実行時間に影響するパラメタ（含む、GPU実行形態の変更）を調整することで、シミュレーション時間を20%ほど高速化できる事例を明らかにしている。以上から量子関連技術においても、AT技術適用の必要性が明らかになってきている。

3.2.9.4　来年度計画

来年度においても、ATツール調査を継続する。特に、米国DOEにおけるATツールの機能と適用事例調査を進めるとともに、AIにおけるATツールの機能調査をAI基盤調査Gと連携して行う。また縦串としての観点から、アプリGベンチマークをいくつか取り上げ、AT適用の効果に関する調査を行う。加えて、量子コンピュータ関連技術におけるAT適用調査も継続して行う。

参考文献

[1] 森下ほか、量子コンピューティングへの自動チューニングの適用と評価、情処研報、Vol. 2023-HPC-188, No. 2, pp. 1-7 (2023-03-09)

3.2.10 HPC 利用環境調査

3.2.10.1 調査目的

本サブグループ（HPC利用環境サブグループ)では、ポスト「富岳」時代の次世代計算基盤のあり方を検討・考察することを視野に、高性能計算(HPC: High Performance Computing)システムを構成するシステムソフトウェア・ライブラリの利用状況について調査を行うことを目的とする。今日においては高性能計算システムの利用は多岐にわたり、学術、産業の枠を超え、多様な方法で利用されている。そのため、ポスト「富岳」時代の次世代計算基盤においては、大規模・大容量なデータを扱うシミュレーションや数値計算だけでなく、今日急拡大傾向にあるAI(人工知能：Artificial Intelligence)、ML(機械学習：Machine Learning)、DL(深層学習：Deep Learning)に代表されるデータ分析からの計算需要・ニーズ・要求を考慮しなければならない。また、今日においては、大規模な科学計測機器、IoTセンサなどの多種多様なデータ源から発生するデータを速やかに高性能計算システムに収容し、高性能計算システムでの解析・計算処理に利用したいという状況もある。IoTセンサで取得された観測データを計算シミュレーションの中間結果と比較し、計算シミュレーションの精度を向上させようとするデータ同化(data assimilation)への期待も高い。さらに、わが国が国際社会と協調して積極的に取り組んでいる、持続可能な開発目標(SDGs: Sustainable Development Goals)から派生する社会課題解決の視点からも、ポスト「富岳」時代の次世代計算基盤は検討・考察されなければならない。加えて、わが国においては、サイバー空間(仮想空間)、フィジカル空間(現実空間)を高度に融合させることにより、経済発展と社会課題の解決を両立する人間中心の社会を目指すSociety 5.0がわが国の目指すべき未来社会として科学技術政策の一つとして掲げられており、ポスト「富岳」時代の次世代計算基盤を検討・考察する上で

重要なキーワードともなる。さらに、今日の学術・産業界における研究活動は、ますますグローバル化、ボーダレス化する傾向にあり、地理的に分散して推進される研究開発に携わる研究者らのチームとしての研究活動の生産性向上という視点からも、ポスト「富岳」時代の次世代計算基盤のあり方は検討・考察されなければならないと考える。本グループ(システムソフトウェア・ライブラリ調査グループ) ではポスト「富岳」時代の次世代計算基盤について、広範なSDGs・Society 5.0の実現に向けた課題解決のためのプラットフォームを実現すべく、高度なデジタルツインによる研究DX基盤となり得る次世代システムについて、サイエンス・産業・社会のニーズも考慮しながら、それを実現可能なシステム等の選択肢を提案することを目的とする。当該目的のために、本サブグループでは、次世代計算基盤のアーキテクチャ、システムソフトウェア・ライブラリ、アプリケーションの調査と要素技術の検討を行い、技術的課題や制約要因を抽出しつつ、次世代計算基盤に求められる性能・機能要件を明らかにする。また、我が国として独自に開発・維持するべき技術に関しても検討する。

3.2.10.2 調査内容

本サブグループでは、上記目的達成を鑑み、データサイエンス、AI技術、シミュレーションの融合、リアルタイムデータ処理、データ駆動型科学の計算ニーズ・要求を収容可能な高性能計算(HPC)システムのアーキテクチャについて調査を行うとともに、わが国の科学技術の発展・高度化に寄与できる次世代計算基盤の中核となりうる計算基盤技術、ネットワーク技術、データ基盤技術の利用環境について調査検討を行う。初年度となる2022年度は、上記目的遂行のために、アプリケーション調査グループ、およびシステムソフトウェア・ライブラリ調査グループ内のサブグループと連携し、システムソフトウェア・ライブラリの調査指標を検討しつつ、わが国のHPCI構成拠点および海外の主要計算機センタにおけるHPC利用環境、とりわけ、システムソフトウェア・ライブラリ等のソフトウェアスタックについての現状分析や、利用実態・状況について調査を開始した。

3.2.10.2.1 調査方針

具体的な調査方針を記す。初年度である2022年度は、本グループが対象とするシステムソフトウェア・ライブラリが国内主要計算機センタにおける高性能計算システム(スーパーコンピュータ)に導入されているかの全体概要把握とともに、その利用者からのシステム・ライブラリに関する利用方法の実態を把握することをねらった。また、計算機システム管理者向けに対してシステムソフトウェア・ライブラリの利用状況・利用実態の調査を行うことを計画した。これにより、利用者の利用実態と計算機システム管理者の間のギャップを浮き上がらせ、ポスト「富岳」時代の次世代計算基盤におけるシステムソフトウェア・ライブラリのあり方を検討・考察するための調査報告書の作成を試みるアプローチを選択している。

3.2.10.2.2 調査方法

上記の調査方針に基づき、本サブワーキンググループでは、2022年度下記を実施した。

- HPCI 構成機関におけるシステム・ソフトウェア導入状況の整理
- アプリ利用者を対象としたアンケートによるシステム・ライブラリの利用状況・利用実態調査
- 計算機システム管理者に対する口頭ヒアリング項目作成

3.2.10.3 調査結果

以下、2022年度末までの本サブワーキンググループ内の利用状況調査をまとめる。

3.2.10.3.1 HPCI 構成機関におけるシステム・ソフトウェア導入状況の整理

本調査項目では、理化学研究所計算科学研究センター、北海道大学情報基盤センター、東北大学

サイバーサイエンスセンター、筑波大学計算科学研究センター、東京大学情報基盤センター、東京工業大学学術国際情報センター、名古屋大学情報基盤センター、京都大学学術情報メディアセンター、大阪大学サイバーメディアセンター、九州大学情報基盤研究開発センター、海洋研究開発機構地球情報科学技術センター、統計数理研究所統計科学技術センター、産業技術総合研究所の運用する24の高性能計算システム（スーパーコンピュータ）を対象とし、ウェブ等で公開されている情報をもとに、システム・ソフトウェア、ライブラリの導入状況を表(全体の表は中間報告の段階では掲載しない)に整理した。

本報告書執筆時点においては、当該整理情報に基づき、わが国における計算機システムのシステムソフトウェア・ライブラリ導入実績情報を分類・可視化する作業を推進中である。下記に現時点での調査結果の一部を記す。

本項目による調査結果(中間報告であり、最終報告時点までに更新される可能性があることに注意されたい)に基づけば、上記を対象とした24の高性能計算システムでは、システム基盤を構成するOSとしての導入実績はRed Hat Enterprise Linuxが調査対象となったシステムの約半数で導入されていることがわかる。CentOS、RockyなどのRedHat系Linuxディストリビューションが調査対象となったシステムにほぼ全て(約92%)に導入されているといえる。ただし、CentOSは2024年6月をもってCentOS Linux7サポートの終了を予定しており、CentOSはLinux7をもって全てのサービスの終了を予定している。このため、今後の調査ではCentOSの代替ディストリビューション候補（RHEL,CentOSとの互換性、安定性などを踏まえた無償版Rocky, Alma, openSUSE, ubuntu などとともに)を有償ディストリビューションも踏まえて動向に注視する。

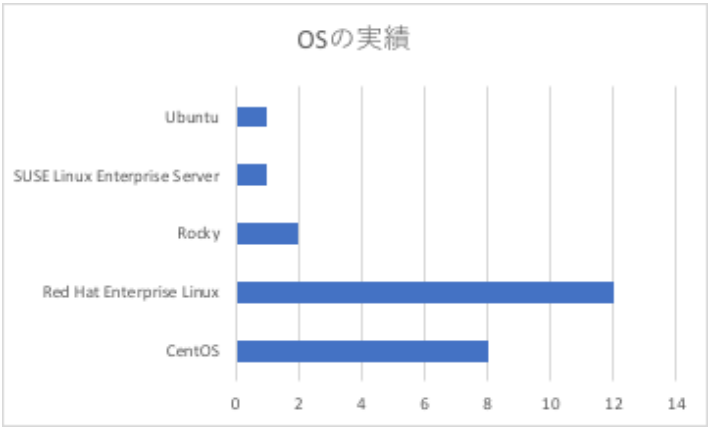


図 3.2.20 HPCI 構成機関所有 24 システムでの OS 導入実績.

メッセージパッシング方式に基づく通信ライブラリMPIについては、多様な実装が利用可能であるが、各種MPIの導入実績を図 3.2.21にまとめた。MPIについては高性能計算システムに導入されるMPIは複数種可能であるため、調査結果の総数が調査対象システム数と一致しないことには注意されたい。この結果から、HPCI構成機関所有システムの半数はIntel社の提供するIntel MPIを導入していることがわかる。次に、オープンソースで利用可OpenMPIを採用しているシステムも調査対象システムの37.5%存在することがわかる。

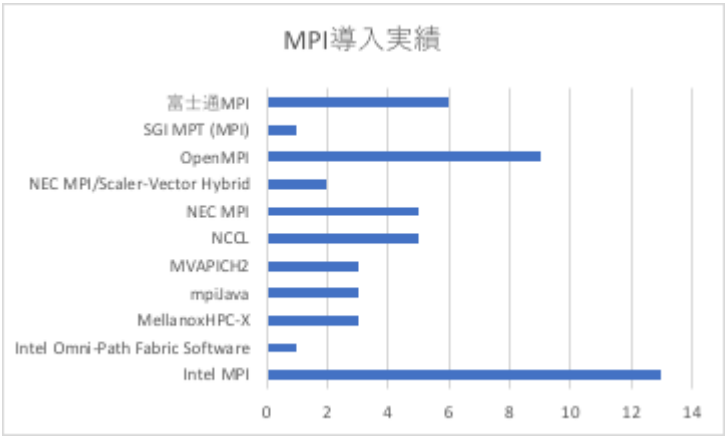


図 3.2.21 HPCI 構成機関所有 24 システムでの MPI 導入実績.

開発環境の中核となるコンパイラ/インタプリタの導入実績情報を図 3.2.22にまとめた。Intelコンパイラについ てはIntelMPIと同様に調査対象システムの半数で導入実績がある。これはコンパイラについても高性能計算システムを構成するプロセッサのアーキテクチャに依存する部分が大きいと考えられる。一方で、オープンソースベースのGNUコンパイラの導入も半数以上で見られる。このことは高性能計算システムの利用者全てが性能を追求する利用者とは限らないことが推測される。

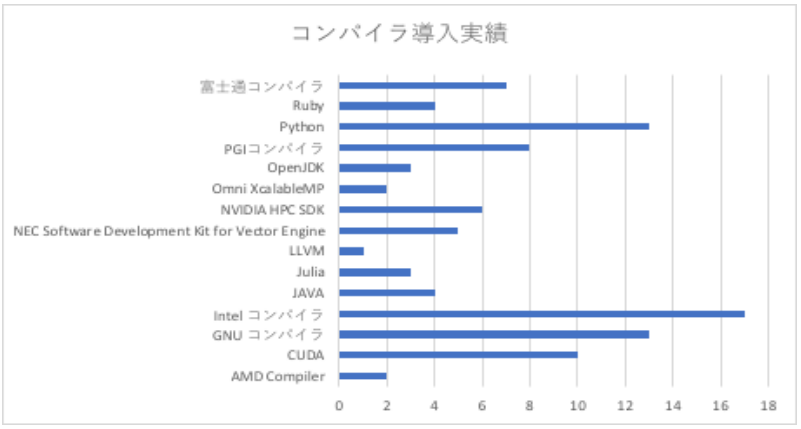


図 3.2.22 HPCI 構成機関所有 24 システムでのコンパイラ導入実績.

今日、AI、データ分析・解析分野における役割・必要性は高い。こうしたことから、図 3.2.23にPython関連ライブラリの導入実績をまとめた。ポスト「富岳」時代の次世代計算基盤においても必要性の高いソフトウェアであると考えられることから導入実績をまとめた。調査対象となる計算機システム数(24)、および、今日のAI、データ分析分野への期待と関心を鑑みると、Python関連ライブラリの導入は数多くないとも考えられる。

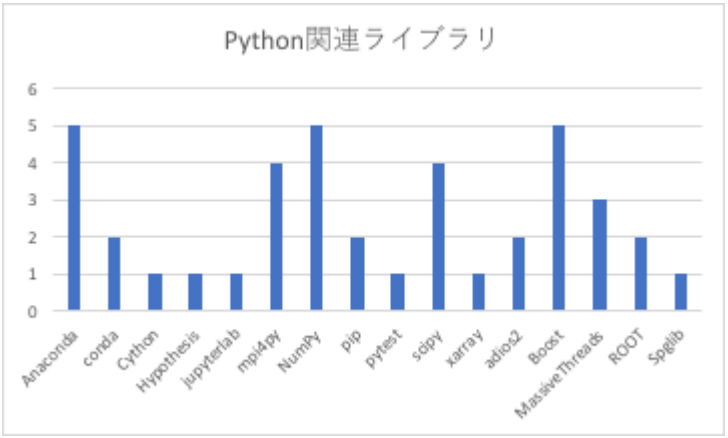


図 3.2.23 HPCI 構成機関所有 24 システムでの Python 関連ライブラリ導入実績.

本調査項目では、上述の通り、調査対象となるシステムのシステムソフトウェア・ライブラリ導入状況を整理、可視化を進めている。これにより、わが国の主要な高性能計算システムにおけるシステムソフトウェア・ライブラリの導入状況については、数値化・可視化は可能となる。しかし、本調査方法においては、例えば、調査対象となり利用者の想定や、管理者によるシステムコンセプトであったり、図 3.2.20および図 3.2.21に示したOSやMPIの導入実績の背後にある導入理由、また図 3.2.22にあるコンパイラ導入にかかわる制約（導入するスケジューラの制約など）や導入理由、Python関連ライブラリの非導入理由などは明確にならない。そのため、本サブワーキンググループでは、次年度(2023年度）にむけ、計算機システムの管理者にむけたヒアリングを計画している。

3.2.10.3.2 アプリ利用者・開発者を対象としたアンケートによるシステム・ライブラリの利用状況・利用実態調査

本調査項目は、利用および開発者視点でのシステムソフトウェア・ライブラリに関する利用状況調査を目的として、本グループが計画したアプリ利用者向け・開発者向けアンケート(【アプリ利用者】次世代計算基盤に係る調査研究事業のためのシステムソフト・ライブラリ利用状況の調査、および、【アプリ開発者】次世代計算基盤に係る調査研究事業のためのシステムソフト・ライブラリ利用状況の調査；母数：30および 31)を通じて実施した。具体的には、高性能計算システムの利用形態に関する希望と利用状況、コンテナ仮想化および仮想計算機技術に対する必要性と利用状況、計算ジョブの入出力データに対する必要性および問題点、インターコネクトのオフロード機能の利用実態と必要性、IoTデバイス等のシステム外データのリアルタイム処理の必要性などについて質問項目を作成し、利用者の現在の利用状況と利用希望のギャップを浮き上がらせることを目的とした調査を実施した。

3.2.10.3.2.1 高性能計算システムの利用形態に関する希望と利用状況

本項目では、高性能計算システムの利用形態として、バッチ・インタラクティブ利用、ジョブ最長実行時間の設定、CLI・GUIの3点について質問した。

下記のバッチ・インタラクティブ利用の質問では、現在高性能計算システムにおいて主流の利用形態であるバッチ利用の満足度について質問した。1.1の質問の結果を見ると、利用者、開発者でバッチ利用に満足しており、インタラクティブ利用の必要性を感じない人は7割を超えている。さらに、バッチ利用とインタラクティブ利用の必要性を比率として質問した結果を見ると、インタラクティブ利用に大きな必要性を感じる人は少数に留まっている。ただし、プログラムの開発やデバッグ、実行結果の可視化等のためにイン

タラクティブ利用が必要との声もあるため、全く必要ないと感じているわけではないと思われる。

質問ID	1.1
質問内容	今日のスーパーコンピューティングシステムはバッチ利用によるものが多くなっております。インタラクティブでの利用形態についてのお考えをお聞かせください。
選択肢	● バッチに満足、インタラクティブ利用の必要性を感じない ● バッチに不満、インタラクティブ利用の必要性を感じる
回答者数	利用者21人、開発者28人

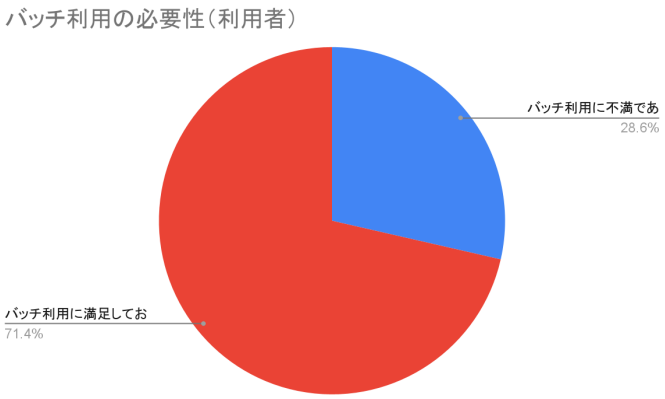


図 3.2.24 バッチ利用とインタラクティブ利用の必要性(利用者).

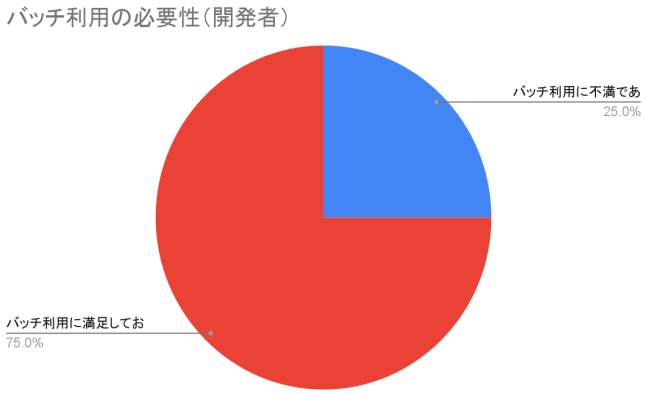


図 3.2.25 バッチ利用とインタラクティブ利用の必要性(開発者).

質問ID	1.2
質問内容	「1.1」の質問と関連しますが、バッチ利用とインタラクティブ利用の必要性を配分する場合、どのような比率となりますか？
選択肢	比率を選択肢とした。
回答者数	利用者22人、開発者26人

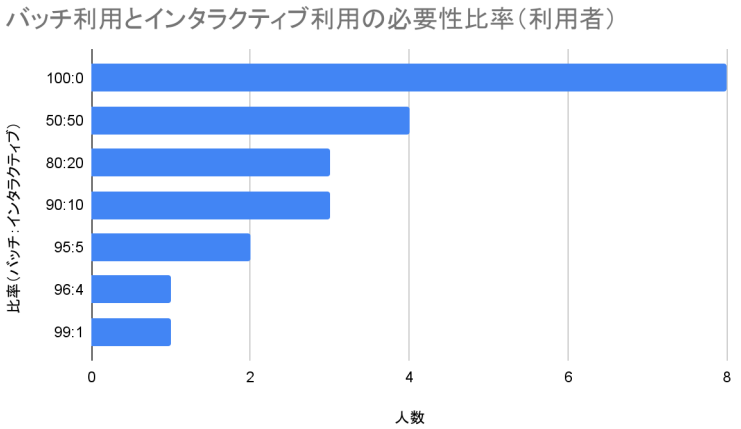


図 3.2.26 バッチ利用とインタラクティブ利用の必要性比率(利用者).

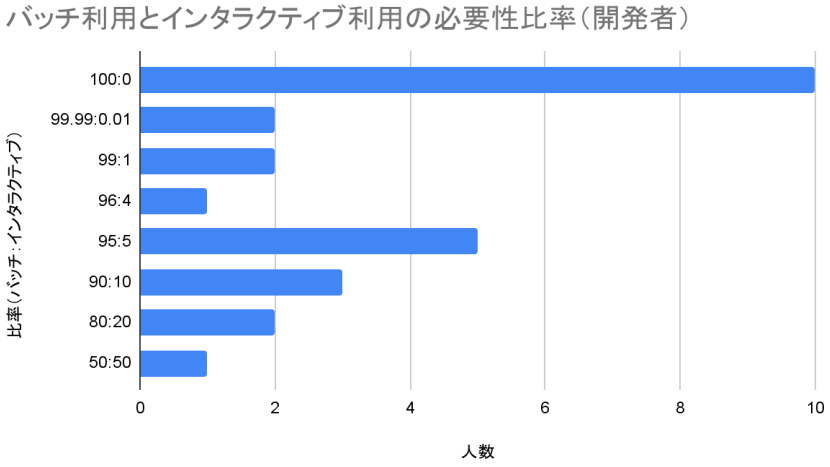


図 3.2.27 バッチ利用とインタラクティブ利用の必要性比率(開発者).

下記のジョブ最長実行時間の設定についてでは、その設定が気になるかどうかについて質問した。2.1の質問の結果を見ると、利用者、開発者の大半はジョブ最長実行時間の設定を気にしていないことが分かる。さらに、少数ではあるが、自身のジョブの待ち時間を減らすために、ジョブ最長実行時間の設定を積極的に導入してほしいとのコメントを寄せていただいた方もいた。しかし、ジョブ最長実行時間の設定を気にする人も約2割は存在するため、優先度は低くなるものの何らかの配慮が必要になると思われる。例えば、限定的に長時間実行を許す制度を導入して欲しいとのコメントが回答の中に存在した。

質問ID	2.1
質問内容	ジョブの実行時間（最長実行時間）について
選択肢	● ジョブの（予定）最長実行時間の設定をすることは気にならない。 ● ジョブの（予定）最長実行時間の設定をすることは気になる。 ● 最長実行時間を求めるようなシステムは利用しない、あるいは、利用していない。 ● その他
回答者数	利用者22人、開発者28人

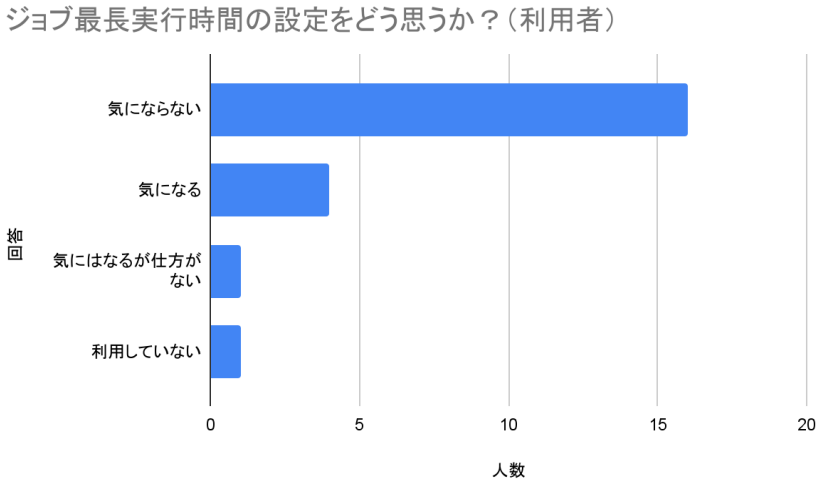


図 3.2.28 ジョブ最長時間設定(利用者).

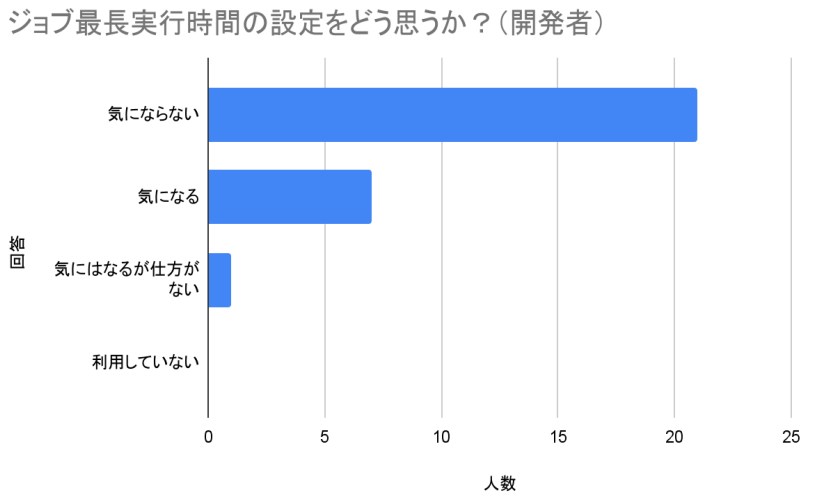


図 3.2.29 ジョブ最長時間設定(開発者).

下記のCLI・GUIについての質問では、CLI利用の満足度について質問した。2.2.1の質問の結果を見ると、利用者、開発者のほとんどはCLI利用に満足している。特に、開発者は回答者の全員が満足していると回答した。このことから、CLI利用が高性能計算システムにおける利用形態の主流であるという今

日の状況は、今後も変化はないと見込まれる。ただし、本アンケートは既存の高性能計算システムの利用者、開発者に対して実施したものであるため、新規の利用者のためにGUIを用意する利点は存在すると思われる。実際に、本アンケートの回答者にも、同様のコメントを寄せていただいた方がいた。GUI利用を導入する場合のコメントとしては、導入による計算負荷の上昇やログインノードの安定性を損なうようなことがないように留意すべきとのことであった。

質問ID	2.2.1
質問内容	今日のスーパーコンピューティングシステムは、ジョブ実行のインターフェースはジョブスケジューラの提供するCLI（コマンドラインインタフェース）となっていることが多くなってます。
選択肢	● CLI に満足しており、GUI の必要性を感じない。 ● CLI に不満であり、GUI の必要性を感じる。
回答者数	利用者23人、開発者29人

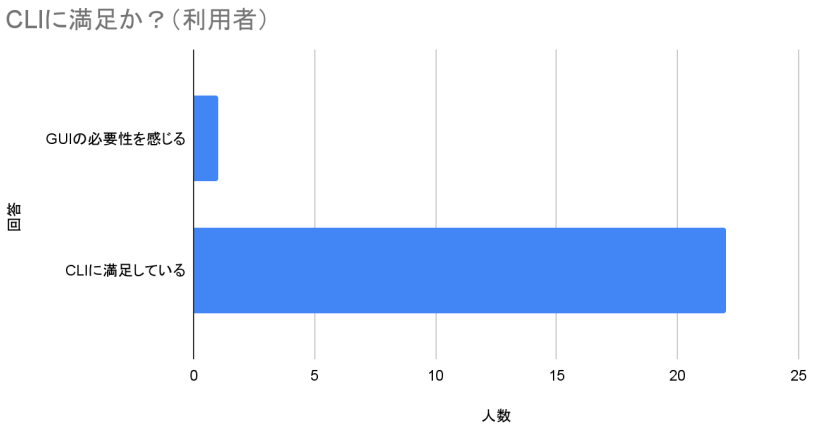


図 3.2.30 CLI と GUI の必要性(利用者).

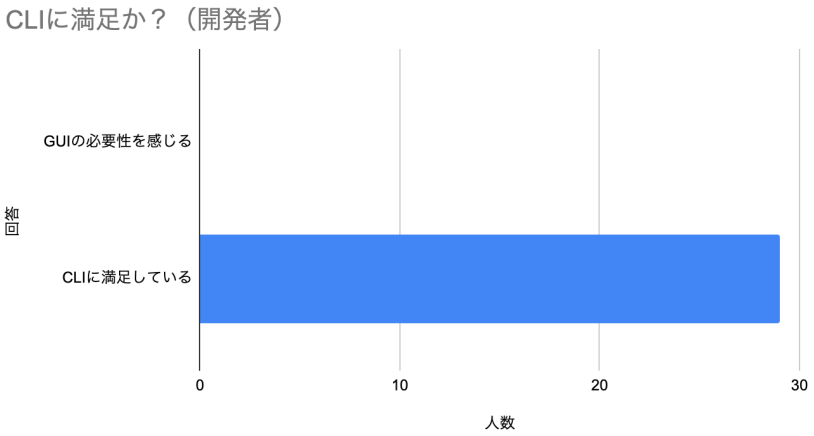


図 3.2.31 CLI と GUI の必要性(開発者).

質問ID	2.2.2
質問内容	「2.2.1」の質問に関して、「GUIの必要性を感じる」を選択いただいた方にお聞きします。CLIとGUIの必要性を配分する場合、どのような比率となりますか？
選択肢	比率を選択肢とした。
回答者数	利用者5人、開発者0人（開発者は全員CLIに満足と回答のため） ※質問ID：2.2.1にて「GUIの必要性を感じる」を選択しなかった本質問の非対象者からも回答が得られたため、その回答数も含む。

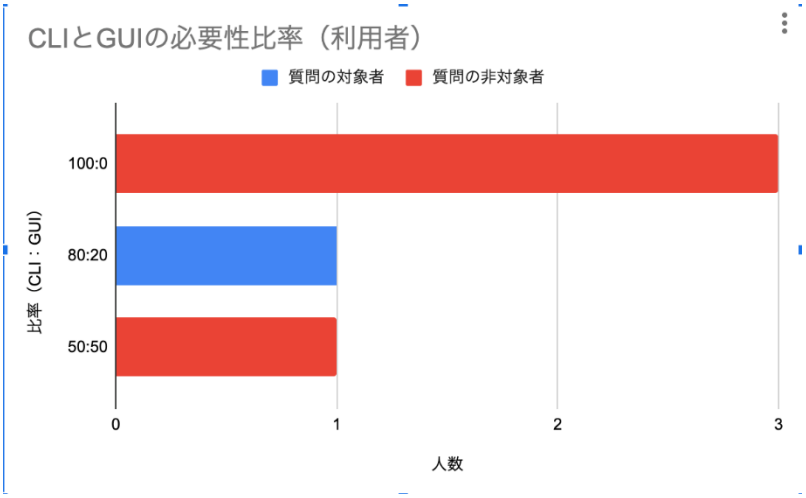


図 3.2.32 CLI と GUI の必要性について比率調査(利用者).

3.2.10.3.2.2 コンテナ仮想化および仮想計算機技術に対する必要性と利用状況

DockerやSingularityなどといった仮想環境には、システムに導入されているものとはバージョンの違うコンパイラやライブラリの利用を可能にし、古いアプリケーションの移植性を担保できるという利点がある。また、第三者が公開するコンテナイメージを利用することで実行環境の開発を高速化可能という利点も存在する。本項目では、下記の通り仮想環境の利用状況について質問した。各グラフは、環境ごとについて回答者が利用したいと答えた割合を集計したものを示す。利用者が利用したい仮想環境の結果を見ると、Docker及びSingularityを利用したいと答えた人はそれぞれ4割を超えている。また、どれも必要ないと感じる利用者は2割強に留まっている。一方で、開発者が利用したい仮想環境の結果を見ると、どれも必要ないと感じる開発者が半数を超えている。このように、利用者と開発者の間には、仮想環境の必要性に対する認識にギャップがあると思われる。ただし、開発者でいずれかの仮想環境を利用したいと回答した方からは多くの肯定的なコメントをいただいたのに対して、必要性を感じないと回答した方からのコメントは少なかったため、単に開発者自身の開発するアプリケーションには必要がないという認識なだけの可能性がある。

質問ID	2.3.1
質問内容	今後のスーパーコンピューティングシステムでの仮想環境についての考えをお聞かせください。
選択肢	下記の項目の重複を許して選択できる形で提示した。 ● Docker を利用したい ● Singularity を利用したい ● Kubernetes を利用したい ● 必要性を感じない ● その他
回答者数	利用者16人、開発者22人

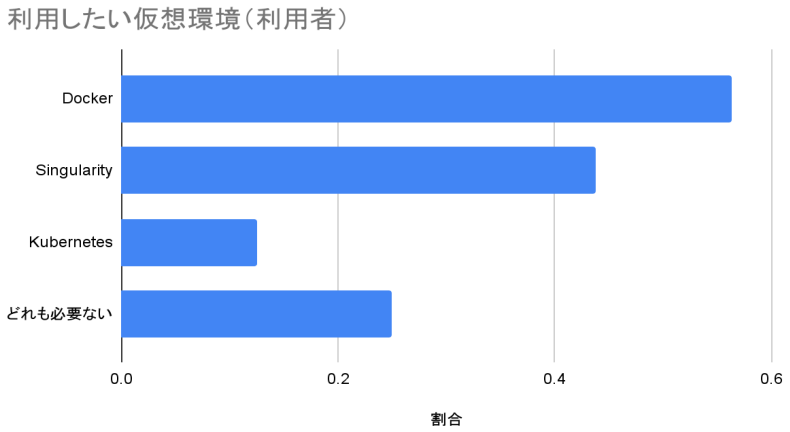


図 3.2.33 利用したい仮想環境(利用者).

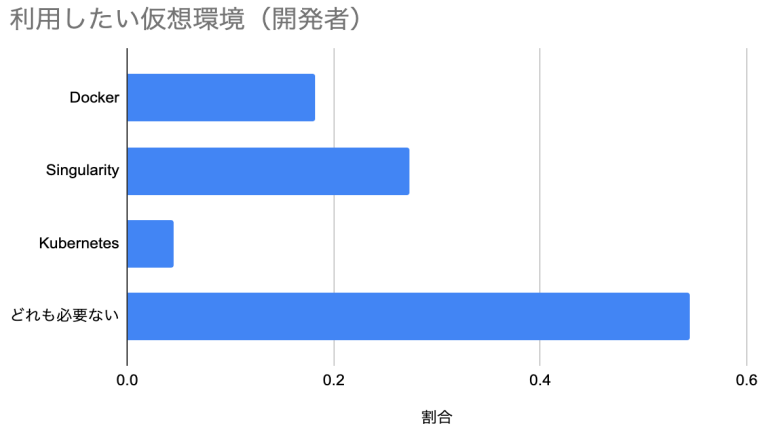


図 3.2.34 利用したい仮想環境(開発者).

3.2.10.3.2.3 計算ジョブの入出力データに対する必要性および問題点

ジョブスケジューラに付随する機能として、ジョブ実行前と後にファイルを計算ノードに転送する機能としてステージング機能がある。実行に必要なデータをジョブ実行開始前に転送し、ジョブ実行後に実行結果

をログインノード等に転送する機能であり、大量のデータを用いるアプリケーションでは必要となる機能である。このステージング機能についてアンケートを実施したところ、利用者の2割程度、開発者で3割程度、必要であると回答している。実際にはアプリケーションの中で必要なファイルを転送する仕組みを用いているものもあり、ジョブスケジューラの機能として利用していない場面があり、利用していない利用者が多いと考えられるが、多様なアプリケーションがHPCシステムで動作することを考えた場合、必要であり、またステージング機能への理解がまだ浸透していないことから利用していないことも考えられ、講習会などで利用者への理解を求めていくことも考えられる。

質問ID	2.4.1
質問内容	利用されるシステムにステージング機能（計算が実行される計算ノードのローカルストレージにデータを移動させる機能）が備わっているものもあります。ご自身の計算に対するステージング機能の必要性について伺います。
選択肢	● 必要であり利用したい ● 不要であり、利用の必要性を感じない
回答者数	利用者22人、開発者28人

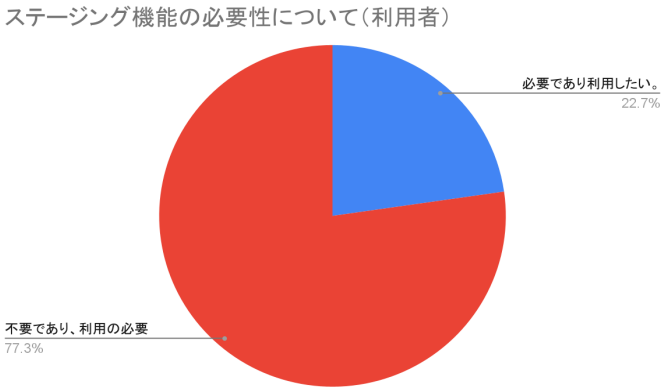


図 3.2.35 ステージング機能の必要性について(利用者).

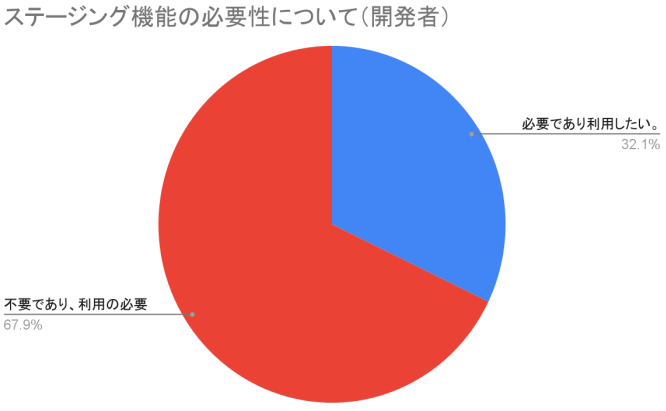


図 3.2.36 ステージング機能の必要性について(開発者).

3.2.10.3.2.4 インターコネクトのオフロード機能の利用実態と必要性

システムが具備するインターコネクトの中には、本来計算ノードが処理していた通信機能の一部をネットワークアダプタやスイッチにオフロードすることで、通信を高速化するソフトウェアやライブラリがある。例えば、集団通信の処理をInfiniBandスイッチにオフロードするNVIDIA Scalable Hierarchical Aggregation and Reduction Protocol (SHARP) が挙げられる。本項目では、下記の通りインターコネクトにおける通信機能のオフロード機能の利用経験及びその機能の必要性について質問をした。利用経験の結果を見ると、利用者、開発者での利用は3割を下回っており、現状として通信機能のオフロード機能の利用が普及しているとは言い難い。開発者からのコメントを確認すると、通信機能のオフロード機能自体に詳しくないとのコメントが多く見られたため、そもそも存在自体の理解が浸透していないと考えられる。そのため、システム側ですでに当該機能が有効化されており、実は利用した経験を有するが、そのことに気づいていないユーザが一定数存在するとも考えられる。一方で、必要性を感じる利用者、開発者は約7割を超えており、通信機能のオフロード機能に対する期待感は伺える。そのため、講習会などで利用者へ説明していくことが重要となると考える。

質問ID	3.3.1
質問内容	システムが具備しているインターコネクトの中には、通信機能の一部をオフロードして通信の高速化をアクセラレートするソフトウェアやライブラリがありますが、利用したことがありますか？
選択肢	● 利用している ● 利用していない
回答者数	利用者22人、開発者27人

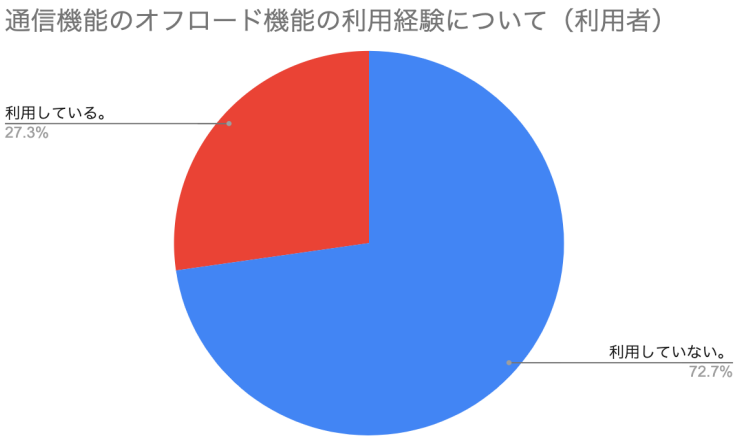


図 3.2.37 インターコネクトへのオフロード機能の利用状況(利用者).

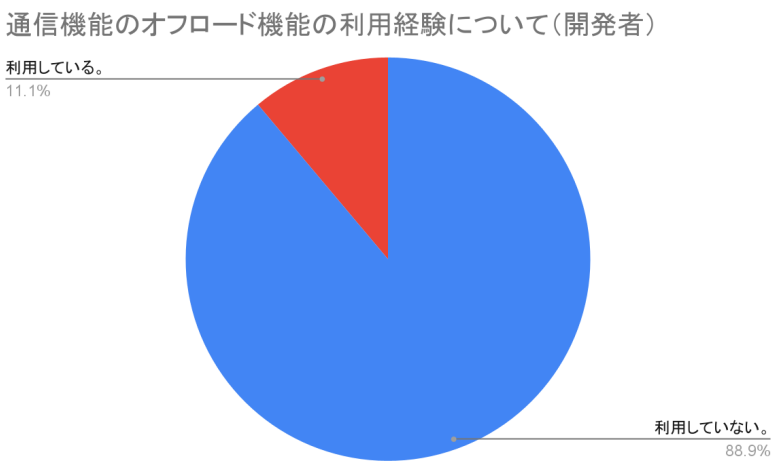


図 3.2.38 インターコネクトへのオフロード機能の利用状況(開発者).

質問ID	3.3.2
質問内容	通信機能の一部をオフロードして通信の高速化をアクセラレートするソフトウェアやライブラリについての必要性についてお伺いします。
選択肢	● 必要、利用している ● 不要、利用の必要性を感じない
回答者数	利用者20人、開発者24人

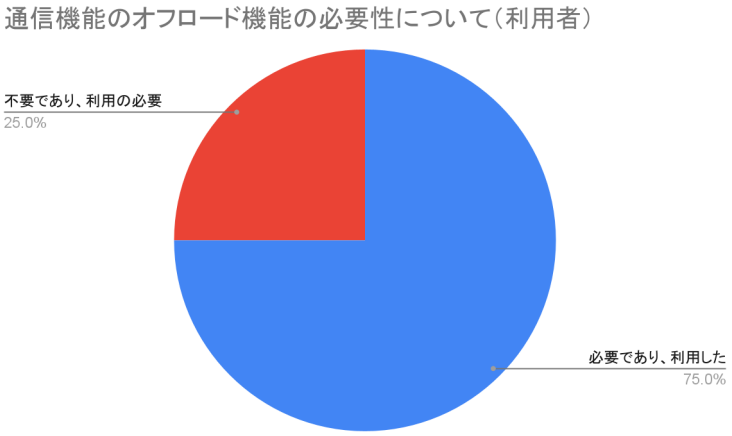


図 3.2.39 インターコネクトへのオフロード機能の必要性(利用者).

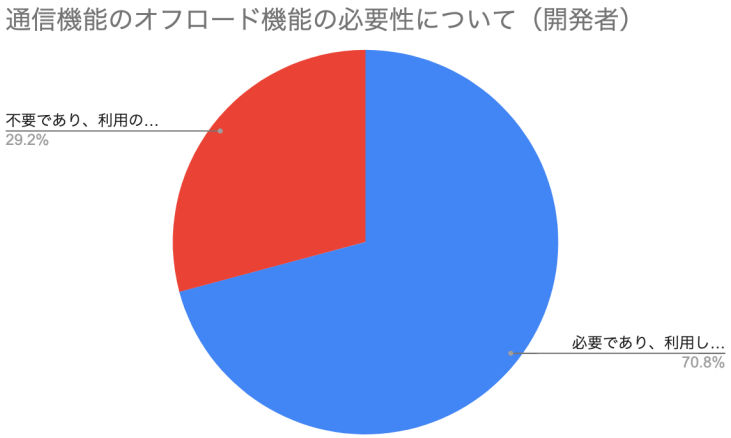


図 3.2.40 インターコネクトへのオフロード機能の必要性(開発者).

3.2.10.3.2.5 IoT デバイス等のシステム外データのリアルタイム処理の必要性

質問ID	4.1
質問内容	開発されるアプリケーションで、今後リアルタイムに生成されるデータ (IoTセンサのデータ等) を扱うシステムの必要性について
選択肢	● すでに使っている ● 必要性がない ● 必要性があると考えている
回答者数	利用者21人、開発者26人

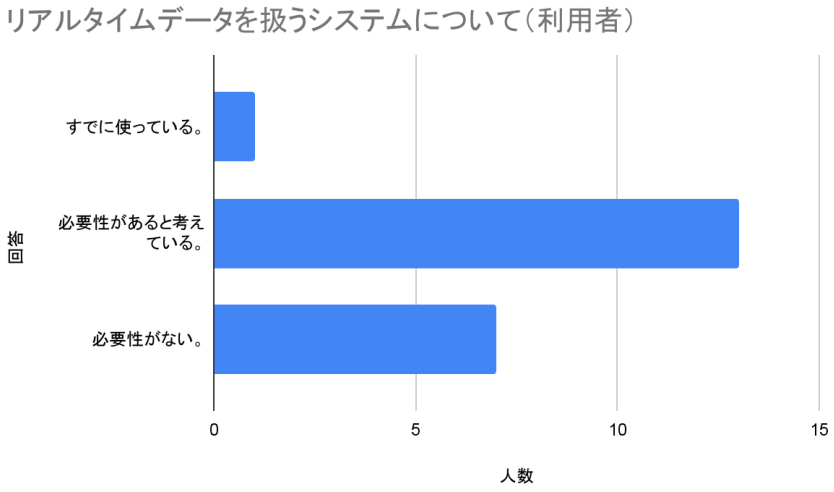


図 3.2.41 リアルタイムに生成されるデータを扱うシステムの必要性(利用者).

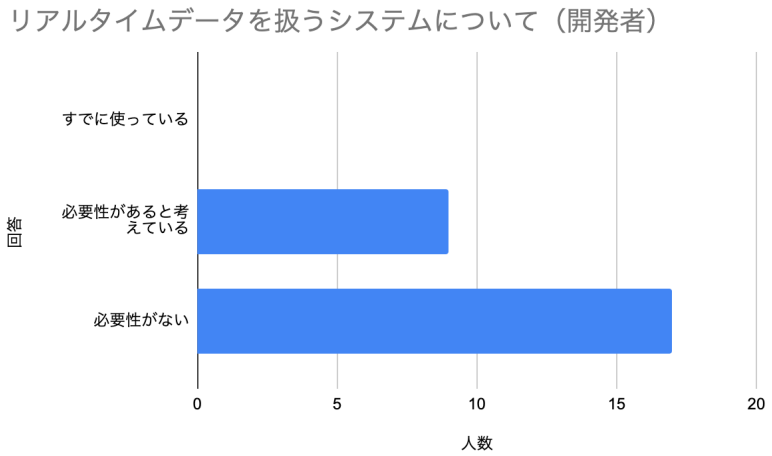


図 3.2.42 リアルタイムに生成されるデータを扱うシステムの必要性(開発者).

質問ID	4.2
質問内容	開発されるアプリケーションで、実行中にWWWに公開されているDBなどへアクセス、またはダウンロードなどを可能とするようなシステムの必要性について
選択肢	● すでに使っている ● 必要性がない ● 必要性があると考えている
回答者数	利用者22人、開発者29人

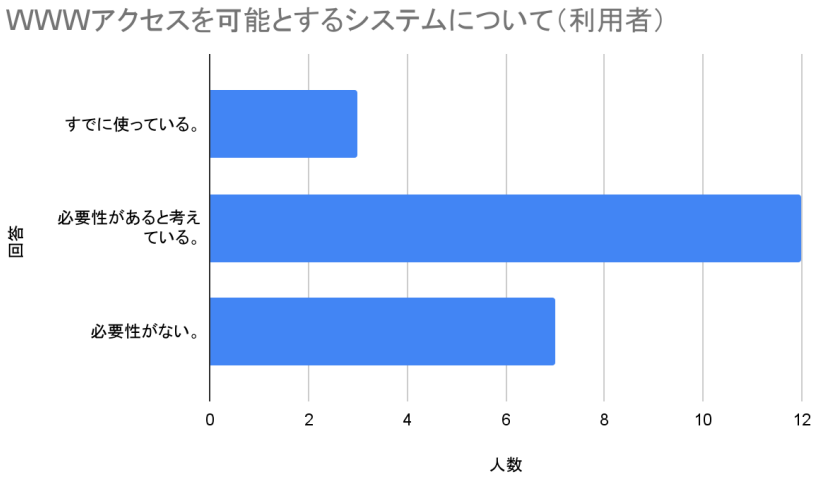


図 3.2.43 WWW アクセスを可能とするシステムの必要性(利用者).

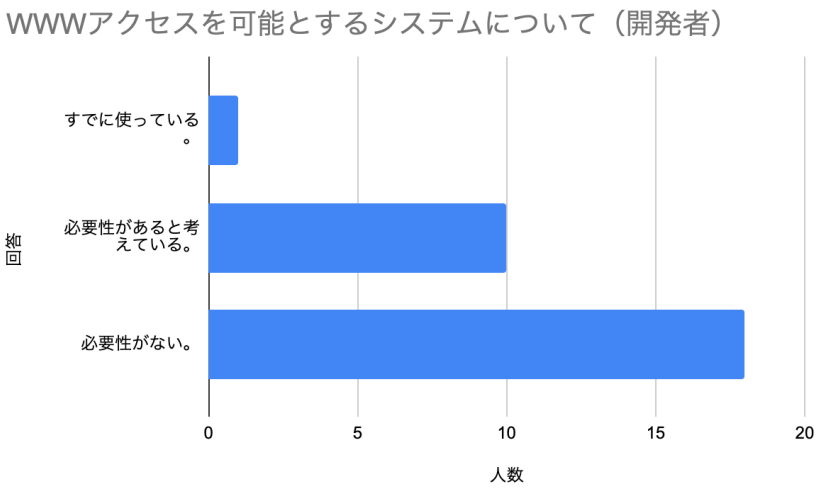


図 3.2.44 WWW アクセスを可能とするシステムの必要性(開発者).

リアルタイムデータの扱いについては、すでに使っていると回答した利用者、開発者は少数に留まっており、現状としてリアルタイムデータを取り扱う知見が揃っているとは言い難い。一方で、必要であると考えている利用者は半数を超え、リアルタイムデータ利用への期待感が伺える。リアルタイムデータは、IoTセンサデバイス、防犯カメラ、スマートフォンなど数多くの機器から多数のデータが送られてくることが示唆され、例えばNIIが取り組んでいるSINETStreamのようなセンサデータをPublisher-Broker-Subscriberのアーキテクチャを取るなど、センサデータ収集に適したアーキテクチャへの適用が期待される。現状のHPCシステムでは、特にBrokerといった不特定多数へのアクセスが見込まれる中継サーバからのアクセスを許可する必要があり、特にセキュリティ面での一層の拡充が必要となると予想される。

WWWアクセスについては、すでに使っていると回答した利用者、開発者は少数に留まっているが、必要性があると考えている利用者は半数を超えており、WWWアクセスについても期待感が伺える。WWWアクセスは、例えばNextCloudといったストレージサービスへのアクセス、あるいはJupyter Noteのような WWW画面からの計算機のインタラクティブ利用がある。すでにWWWアクセスインタフェースが用意されているHPCシステムもあるが、利用が多いというわけではない現状が伺える。特に開発者においては、WWW画面ではなくターミナルプロンプトからの利用が一般的であり、単体テストやあるいは試行実行などを手際よく行うには、WWW画面といったキーボード操作のみで完結する操作がしにくい環境を避ける傾向がある。一方で利用者は、出来上がったワークフローやスクリプトなどを実行するのみで、結果をわかりやすく可視化状態で表示されるWWW画面を好む傾向がある。こうした利用形態の差異により、WWWアクセスの利用状況は促進しているとは言い難いが、今後、利用形態の多様化などから利用が見込まれると考えられる。

3.2.10.3.3 計算機システム管理者に対する口頭ヒアリング項目作成

3.2.10.3.1、 3.2で上述したが、アンケート調査ではシステムの導入状況は公開情報からある程度のシステムソフトウェア・ライブラリ導入状況が可能である。しかし、高性能計算システムに導入されているとはいえ、実際は利用されていないものも数多くあると考えられる。そのような利用状況・実態の調査は、アンケートや質問状等の書面では、回答側の回答データの利用に関する不安であったり、質問の意図が明確に伝わらず、たとえ、回答がえられたとしても正確な利用状況・実態を反映できるとは言いづらい側面がある。そのよう

な懸念と考察から、本サブグループでは、次年度(2023年度)実施に向け、対象となるシステムの想定利用者（想定研究分野、想定計算ニーズ）、利用者による想定利用形態、システム・ソフトウェア導入理由（スケジューラ、コンパイラ、OS、モニタリングソフトウェア等）、当該システムで発生している計算ニーズ・要求の変化動向、新しい計算ニーズ・要望に対する当該システムでのチャレンジ等についてのヒアリング項目の作成を鋭意進めている。

下記に現時点での検討中のヒアリング項目を示す。本サブグループでは、下記ヒアリング項目の選定と精査を行ったのち、対象となるシステムの管理者グループに対してできるかぎり対面(オンラインでの対面もやむなし）をヒアリングしていくことを次年度に計画している。

[検討中のヒアリング調査項目]

システム管理者に向けたヒアリング

3.2.10.3.1および3.2.10.3.2にて調査した結果を踏まえ、更に深掘りすることによってHPCI参画機関におけるHPCI利用環境における利用動向の把握によりポスト「富岳」時代の次世代計算基盤整備に必要な技術と環境を明確にする。

(1) インストール・導入ソフトウェアについて

3.2.10.3.1にてHPCI参画機関の各センタに導入しているシステムソフトウェア・ライブラリについてまとめた。これらは各センタの公開情報（web等）を基にカテゴリ毎に整理しているが、詳細まで把握できていないと考えられる。

このため、各カテゴリに対する各センタの動向を詳細に再調査必要と考え、想定利用者（想定研究分野、想定計算ニーズ）、利用者による想定利用形態、システム・ソフトウェア導入理由（スケジューラ、コンパイラ、OS、モニタリングソフトウェア等）等や実際の利用状況について（使用履歴が採取されていない場合は定性的な確認に留める）調査を継続予定である。

(2) 3.2.10.3.2アンケート結果の掘り下げ

3.2.10.3.2においてアプリ開発者、アプリ利用者向けに行った「システムソフトウェア・ライブラリの利用状況の調査」に対して実施したアンケート結果を掘り下げ、発生している計算ニーズ・要求の変化動向、新しい計算ニーズ・要望に対する当該システムでのチャレンジ等を明確にする調査を継続予定である。

- スケジューラについて

  既存利用形態（バッチ、インタラクティブ）、新しいサービス形態に関する考えを確認。ジョブの作成（記述性）やGUI（webからのジョブ投入、参照、削除操作の必要性等についても確認）等の利便性、スケジューラに具備されるべきスケーラビリティ（ジョブNode数、同時投入ジョブ数、同時滞留ジョブ数など）への要件や、スケジューリング機能（高度化する連成シミュレーションにおけるドメインコンピューティングへの対応、データのステージング機能、クラウドバースティングへの対応、仮想基盤への対応など）についての各センタでの考えについてヒアリングおよび、議論する。

- 仮想基盤について

  各センタの仮想基盤への対応、検討状況や利用者ニーズを確認する。

- リアルタイム処理についての対応

  3.2.10.3.2アンケート結果において「リアルタイム分析」を「すでに行っている」「今後必要と考えている」と回答が60%となっており、今後さらなるニーズが高まると予想される。リアルタイム分析の環境として必要と思われるストレージシステム・ネットワークへの各センタの対応、検討状況を確認する。

- ワークフローについて

  複雑化する社会課題への解決に向けて多くの異なる分野のシミュレーションの連携が多くなると予想できる。また、高度化する連成シミュレーションを背景として、3.2アンケート結果においてワークフロー機能について「今後利用が必要と考えている」との回答を得ている。ワークフロー技術動向の調査を踏まえて各センタの対応、検討状況を確認する。

- アプリケーション性能について

  古くはOSジッターから始まり、現在ではシステム・ノードアーキテクチャから発生するアフィニティやインターコネクトのトポロジによるアプリケーション性能への影響やアクセラレータ搭載システムの最適な利用方法について、各センタでの対応状況や対応検討状況を確認しアプリケーション性能に影響する技術等への対応動向を確認する。

3.2.10.4 来年度計画

2023年度においては、3.1HPCI構成機関におけるシステム・ソフトウェア導入状況の整理において整理したシステムソフトウェア・ライブラリ調査表の拡充を測る。具体的には、現時点では公開情報に記載されていないシステムソフトウェア・ライブラリの見直しを行い、引き続き導入情報の整理を行う。さらに、3.2 アプリ利用者・開発者向けアンケートの結果については、回答数が多くなかったことも考慮にいれつつさらに掘り下げ、発生している計算ニーズ・要求の変化動向、新しい計算ニーズ・要望に対する当該システムでのチャレンジ等を明確にする調査を継続する。同時に、3.3に示した管理者ヒアリングを行い、利用者の現在の利用状況と利用希望の合致点・不一致点を明確にする。これにより、どのようなシステムソフトウェア・ライブラリが現状利用されており、また、本グループ内の他サブグループの調査との情報交換を行いつつ、ポスト「富岳」時代の次世代計算機を鑑みた場合、どのようなシステムソフトウェア・ライブラリが利用者にもとめられているのか等を利用者および管理者視点の実態から継続的に調査を推進していくことを計画している。

3.3 まとめと今後の計画

以下に各サブグループの調査結果をまとめる。

- コンパイラ・プログラミングモデル：「富岳」におけるコンパイラや開発ツールに対する振り返りを行った。アーキ G やベンダと連携して候補アーキテクチャ毎に利用可能なプログラミング環境（コンパイラ、プログラミング言語、プログラミングフレームワーク、デバッガ等）を調査。
- スケジューラ・ランタイム：各 HPCI システムや Top500 上位のスパコンで利用されているスケジューラ・ランタイムの調査（半数以上のセンタが SLURM 利用）。ジョブスケジューリングの課題の調査。その他、ランタイム（ワークフロー管理ツール、プロファイラ等）の調査。
- 通信ライブラリ：ノード間・ノード内通信、ワークフロー、低レベル通信ライブラリの現状と利用されているシステムソフトの調査。

- I/O・ストレージ・ファイルシステム：デバイス・インタフェース・ネットワーク（HDD, SSD, PCIe, CXL）の技術動向の調査。HPC で利用されている代表並列ファイルシステム（Lustre, DAOS 等）に関して調査。富岳ストレージの課題の洗い出しと改善策の提案。
- 数値ライブラリ：「富岳」における数値計算ライブラリに対する振り返りを行なった。数値計算ライブラリ（密線形代数、疎反復ソルバ、疎直接ソフバもしくは階層行列ソルバ、高速フーリエ変換、プログラミングフレームワーク、高姿勢性、グラフ、DSL、テンソル等）の調査。
- AI フレームワーク：LLM で利用されている Transformer に関する調査と各アーキテクチャに対する深層学習向け数値計算ライブラリの調査。
- OS・仮想化・クラウド連携： 既存スパコンにおける OS 利用状況と軽量 OS に関する調査、HPC 分野における仮想化技術の動向調査、アプリケーションユーザからの OS、仮想化技術、クラウド連携への要望調査。
- その他の技術調査：各システム・ソフトウェア分野を跨ぐ自動チューニング技法やセキュリティに関する技術調査。
- HPC 利用環境：HPCI 構成機関におけるシステム・ソフトウェア導入状況の整理、アプリ利用者・開発者を対象としたシステム・ソフトウェア利用実態調査、計算機システム管理者に対する口頭ヒアリングのための質問項目書の作成。

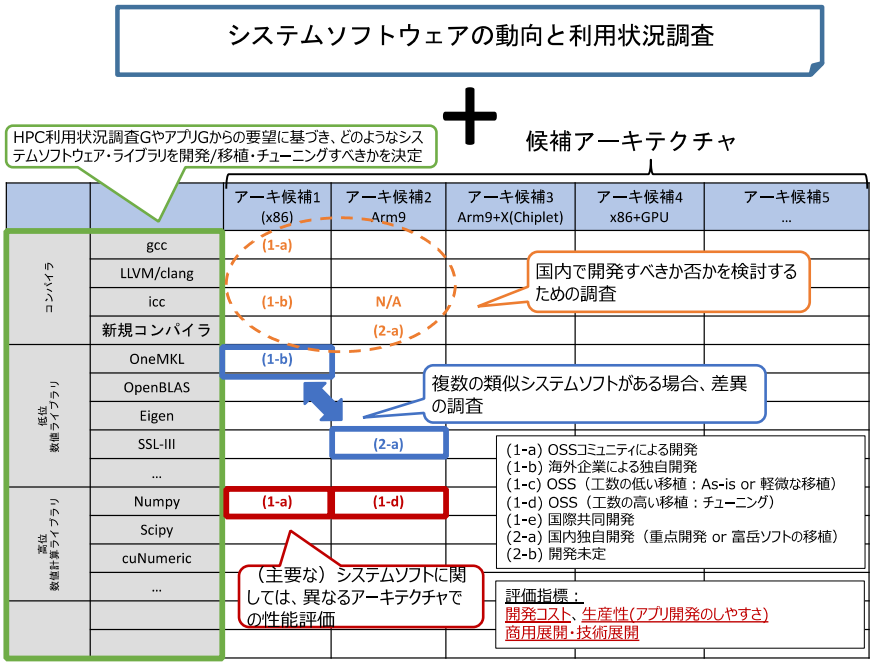


図 3.3.1 システムソフト G の最終報告書の構想案

今後の計画として、【調査項目1】の残りの調査と【調査項目2&3】の実施し、またFSアプリGの枠を超えてより広くシステム・ソフトウェア利用状況を把握するためにシステムログの解析を実施、さらに、必要に応じてアプリGへのアンケートを踏まえてアプリ側の要望を詳細に調査するために各サブグループで個別のヒアリングを実施予定である。最終報告書では、これらのシステム・ソフトウェアの動向と利用状況調査の結果をまとめるとともに、各候補アーキテクチャに対してソフトウェア開発戦略をまとめる予定である。