資料1-2
科学技術・学術審議会　人材委員会
科学技術人材多様化WG（第4回）
令和7年6月30日

大学における コンピュータ技術 支援人材の必要性

富田賢吾
東北大学 理学研究科 天文学専攻
tomida@astr.tohoku.ac.jp

自己紹介

名前：富田 賢吾（とみだ けんご）

現職：東北大学理学研究科天文学専攻 准教授

経歴：京都大学（学部）→総合研究大学院大学/国立天文台（大学院）
→Princeton大学（PD）→大阪大学（助教）→現職

研究１：星・惑星形成過程の数値シミュレーションによる研究
研究２：宇宙物理学向け公開シミュレーションコードAthena++の開発
学生時代からスーパーコンピュータによるシミュレーション研究に従事。
「富岳」を含む多様なスーパーコンピュータの利用経験あり。

今回はソフトウェア開発に関わる人材についてお話しさせて頂きます。

趣旨

- 今日の研究現場では（理系に限らず文系であっても）あらゆる場面でコンピュータ技術が不可欠。
- コンピュータ技術の専門化・先鋭化が進んでおり、プロフェッショナルではない研究者の手に負えないレベルになっている。
- 共通研究基盤としてのコンピュータ技術を提供・支援する人材の需要が高まっており、これを研究機関として整備・提供する必要がある。
- 最先端の研究に必要な技術基盤を担う人材を研究機関内で養成・維持する仕組みが必要（→ 研究に関わるキャリアパスの多様化）。
- ソフトウェアは特定のハードウェアに束縛されないため、機関や地域、世代を超えて高い波及効果と費用対効果を持つ。

→「技術・人材のコアファシリティ化」

研究現場に必要な計算機技術者

- **研究向けソフトウェアエンジニア** (今日は主にこの話)
  - 研究で使用するソフトウェアの開発支援
- アプリケーションエンジニア（デザイン含む）
  - 研究成果の社会実装・製品化・広報普及の支援
- データ可視化スペシャリスト（動画作成・デザイン含む）
  - データ解析の支援、広報普及の支援
- AI/機械学習エンジニア
  - 発展著しいAI・機械学習技術の研究への導入を支援
- データサイエンティスト・キュレーター
  - 研究に必要なデータの取得・管理・解析を支援
- （ネットワーク・セキュリティ管理者）
  - ネットワークやサーバ類の管理 → 現場の負担・セキュリティリスク

研究におけるソフトウェア開発

「研究用ソフトウェア」と言ってもその用途は多岐に渡る:
(A) スーパーコンピュータを利用した数値シミュレーション
(B) 望遠鏡・分析機器などのデータ解析
(C) 人工衛星・実験装置などの制御・運用
(D) 装置・建造物の設計・解析

「研究用ソフトウェア」の入手・利用方法も分野や対象によって様々:
(1) 既存製品を購入する / 特定企業に発注する。(例:Ansys Fluent)
(2) 共同利用機関などで提供される業界標準コードを使用する。(例:OpenFOAM)
(3) コミュニティで開発された公開コードを利用する。(例:Athena++)
(4) 個人～少人数(研究室レベル)でコードを開発する。

(a) ソフトウェアの中身は触らず提供されたものをそのまま使用する。
(b) ソフトウェアの一部または全部を改修・開発して使用する。

今日は(A)数値シミュレーションについて(2)-(4)(b)の形態を想定してお話しします。

ソフトウェア開発の専門性

数値シミュレーションソフトウェアの開発にも幾つかの階層がある:

1. 解くべき問題の設定(例:流体力学)　　分野毎の専門性

2. 解法(アルゴリズム)の選定　　計算機科学/数理科学の専門性

3. ソフトウェア実装(プログラミング)　　プログラミング技術の専門性

4. ソフトウェアの並列化・高速化　　＋アーキテクチャ/システムの専門性

多くの研究者は自身の分野の専門家ではあっても計算機の専門家ではない
→ 1(せいぜい2まで)の専門性までしか持ち合わせていない。

3-4には必ずしも1-2の個別の分野の専門性は必要ではない(あった方が良いが)。
→ ソフトウェア開発技術の専門家がいれば研究全体の質・量を向上させられる。

これは実験装置・分析技術の専門家の必要性と同じ構図 → コアファシリティ化

必要なソフトウェア技術

- シミュレーションソフトウェアの高速化
  - 現代的なCPUの性能を引き出すにはCPUの性質を理解した開発が必要。
  - データ構造の最適化（キャッシュ・並列化）
  - SIMDなどCPUの高速化機能の利用
  - CPUアーキテクチャごとの対応(x64, Arm (e.g. 富岳), RISC-V, etc…)
- 並列化(特に大規模)
  - スーパーコンピュータの性能を引き出すには数万並列以上の大規模並列が必要。
  - 分散メモリ(MPI, libfabric) + 共有メモリ(OpenMP, pthread) 並列
  - 大規模並列で性能を引き出すにはネットワークの知識も必要。
- GPU等演算加速器への対応
  - 最大規模の計算機は(「富岳」後継機含め)ほぼ全て演算加速器を搭載
  - 専用言語(CUDA/HIP)、指示文(OpenACC/OpenMP)、ライブラリ(Kokkos等)
  - 単に書き換えれば良いわけではなくアーキテクチャの性質を理解した開発が必要。

研究者の専門性とは全く異なる専門的計算機技術の需要が高まっている。

解決すべき課題

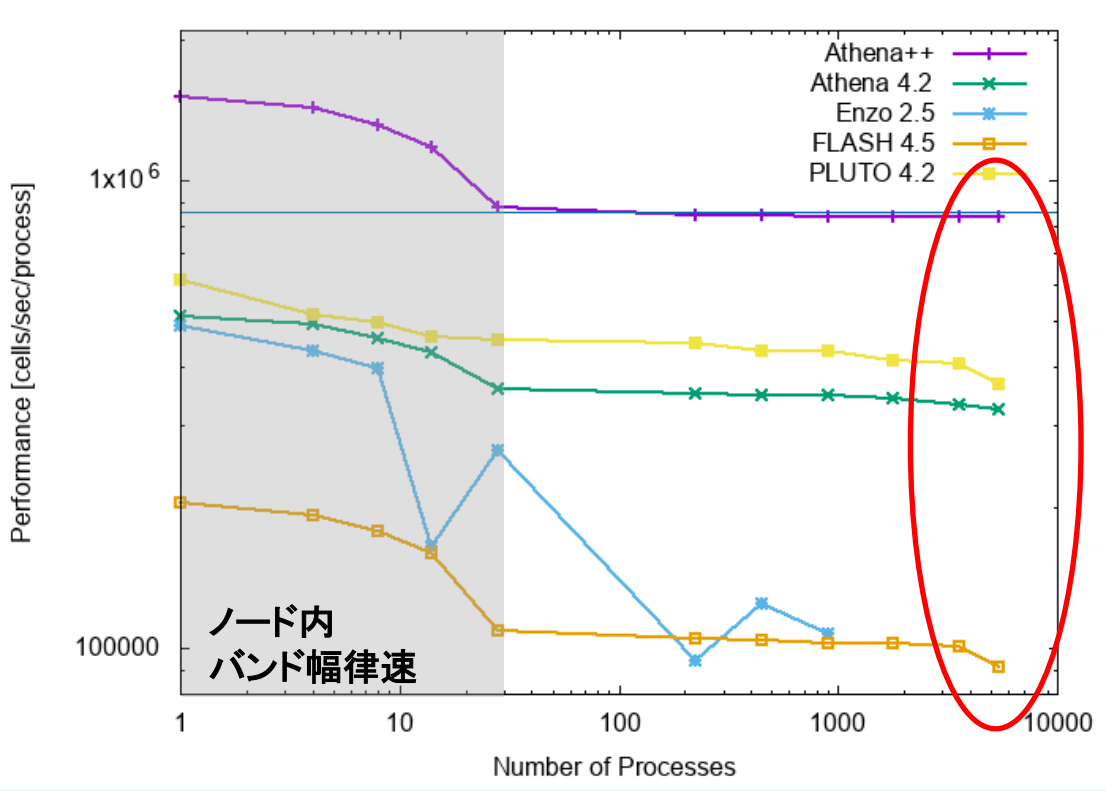
- 殆どの研究者はコード性能について無頓着、無知、あるいは余裕がない。
- 「(正確に)解けること」がまず重要で、速く解けることは優先順位が低い。
- コードの高速化には研究とは異なる知識や技術が必要
  - 最適なアルゴリズムの選定(ここまでは研究者の専門分野依存)
  - 適切なデータ構造の選定
  - 適切な実装
  - 適切な並列化
  - コンパイラや実行時パラメータの適切な設定

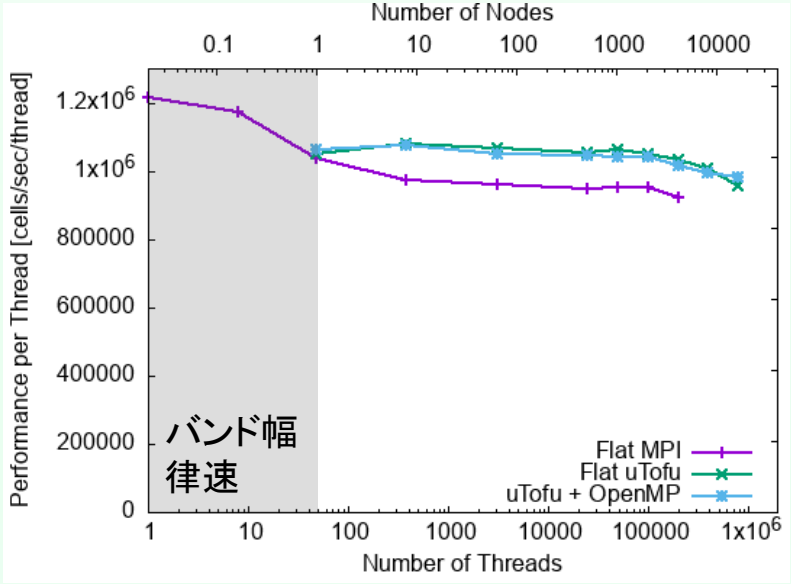
しかし、実装次第で容易に数倍の性能差が出る → 生産性・予算に直結

- 同じ計算機・時間・電力・予算で実行できる計算を拡大可能
- 同じ計算をより小規模な計算機・時間・電力・予算で実施可能 (SDGs)

ポイント:ソフトウェアの高速化は(限界まで突き詰めるとなると大変だが)比較的定型的な処理で効果がある。特に「遅いコード程効果が大きい」。

事例：Athena++





↑ 富岳 48 core / node
← Broadwell 28 core / node
　Flat MPI（少し古いデータ）

宇宙物理学向け公開磁気流体シミュレーションコード (Stone et al. 2020)
2次精度、HLLD+CT MHD、1コア当たり64^3セルの弱スケーリング

- 研究者が書いたコードは必ずしも速くない（＝伸びしろがある）。
- 適切に最適化されたコードは数倍以上高速になる可能性がある。

演算加速器への対応

- 現代は計算機アーキテクチャの一大変革期にある。
- スーパーコンピュータへのGPUを始めとする演算加速器の導入が進んでいる。
  (TOP500リスト中237機が加速器搭載。上位10位中CPUなのは「富岳」のみ。)
- 「富岳」後継機も含め、当面はこの傾向は継続すると考えられる。

- GPUは単純な演算の大量実行に特化 → 従来のコードでは性能が出ない。
- GPUに対応するには各ベンダーの専用言語(NVIDIA CUDA/AMD HIP)を使うか、OpenACC/OpenMP等の指示文ベースのフレームワークを用いる、あるいはKokkosなどのライブラリを使用する必要がある。
- GPUの性能を最大限に引き出すには専用言語に書き換える必要があるが、ソフトウェアの全面的な書き換えが必要な上、特定ベンダーの機材に拘束される。
- いずれの手法であっても、性能を引き出すにはGPUの特性を理解した上でコードを書き換える必要がある。

→ 研究現場にとって演算加速器への対応は急務だが、大きな負担になっている。
⇒ 最先端の計算機の性能を引き出すため、ソフトウェア技術の需要が高まっている。
(c.f. JCAHPC MiyabiではNVIDIA社と提携して移行サポートを実施している。)

大学等での組織イメージ

- 学内での技術共有・水平展開を狙い、全学的な組織とする（学内にソフトウェアコンサルティンググループを設置するイメージ）。
- 数名～のソフトウェアエンジニアと、活動を統括するコーディネータを雇用する。民間からに加え、技術志向の高い院生や研究者から採用することを想定。原則任期無しで民間と同程度の待遇を提示する。
- 積極的・能動的に研究者に対して働きかけ、学内のソフトウェア開発を支援する。
- 最先端のソフトウェア技術を収集・蓄積する。
- 研究者と日常的に交流し、現場で必要な技術と需要を把握する。また、技術交流を軸として研究グループ間を繋ぐハブとして機能する。

人材確保・育成のパスとして

海外の大学や研究所（例：Princeton大学、Max Planck Institute）では研究をサポートするソフトウェアエンジニアを無期で雇用している。

現状の大学では技術専門職（技官）的ポジションは減少している。
また旧来の技官はあまり地位の高い職位だったとは言い難い。
→ 高い専門性を評価し、人材を有効活用できる仕組みを構築する。

アカデミアで働くことを志向しても教員以外のキャリアパスがほぼない。
しかし、研究者・教育者以外の形で研究に関わりたいという人もいるはず。
（論文を書くのは得意ではないが、プログラミングは好き、という人もいる。）
現状そのような人材はいずれ民間企業等に流出するケースが多いが、
キャリアパスを多様化してそのような人材を確保することは大学にとっても＋になるはず。

その他出来そうなこと

- 学生インターン受入による人材育成
- 共同研究のマッチング
- コード開発やシミュレーション、スパコン利用の講習会請負
- ドキュメント・Web整備などオープンソース化・コード公開の支援
- 事例集や技術ノートなどの情報公開
- データの可視化、映像作成等の請負・支援
- 「富岳」後継機など将来の大型計算機のためのターゲットアプリ開発
- 研究用ソフトウェアの開発請負
  - 医学用画像データ分析ツール
  - 宇宙望遠鏡用制御ソフト、公開用データベースとWebシステム
  - 学術調査用のWeb/スマホ用アプリ
- ゴードンベル賞など著名な賞への挑戦

海外の事例

- Princeton University（アメリカ）
  Princeton Research Computingという組織で学内向けの大型計算機を運用するとともに、40名以上の“Research Software Engineer”や類似の職を雇用、各部局の研究を支援。
  https://researchcomputing.princeton.edu/
- University College London（イギリス）
  Advanced Research Computing Centerという組織で
  40名以上の“Research Software Engineer”に加え、
  約10名の“Data Scientist”、10名以上の“Data Stewards”を雇用。
  https://www.ucl.ac.uk/advanced-research-computing/
- Max-Planck-Institute（ドイツ）
  各研究グループで計算機技術者を雇用し研究を支援。

想定Q&A - 1

- 各研究グループで雇用すれば良いのではないか？
  - 研究ソフトウェアは開発期と応用期があり、開発の需要が一定ではない。
    (1グループ内で定常的に開発に関わる仕事があるわけではない。)
  - 研究対象が違ってもプログラミング技術には共通点がある。
  - 多様なソフトウェア開発で培った技術の水平展開による相乗作用が期待できる。
  - 安定した財源による長期雇用・キャリアパス確立の必要性。
  → 研究グループ単位より大きい組織単位で雇用する方が効果的。

- 共同利用機関や計算機センターで雇用すれば良いのではないか？
  - 現状共同利用機関・計算機センターは運用で手一杯で人員に余裕がない。
  - 計算機センターでは特定のハードウェア(スパコン)に紐付けになってしまう。
  - 現状でもそのような支援はあるが、多くの場合は質問対応や「伴走型支援」であり、
    研究者の負担を十分に軽減できていない(また技術力が高いとも限らない。)
  - 質問対応ではなくより積極的に支援するような枠組みがないと活用されない。
  → 十分な人員配置ができればそのような可能性はあるが、運用に注意が必要。

想定Q&A - 2

- 必要に応じて業者に委託すれば良いのではないか？
  - 業者に委託すると高額であり現状のアカデミアの予算規模では難しい。
  - 普段研究ソフトウェアに関与していない会社は必要な技術を有していない。
  - 人材や技術が研究機関・コミュニティに蓄積されない。
  - 開発で習得した技術を他のソフトウェアに展開することが困難。

  （注：研究向けソフトウェア開発がビジネスとして成立する可能性はある。）

- 研究用ソフトウェアは専門的で、共通点が少ない/普通の技術者では不足では？
  - 専門的な知識を要求する部分は当然ある（適切なアルゴリズムの選択等）。
  - 既に動作しているコードを高速化する作業に専門的知識は必要ない。
  - むしろ研究者のプログラミング技術不足で適切な開発ができていないのが現状。

- AIにやらせれば研究者がソフトウェア開発する必要はないのでは？(vibe coding)
  - AIが得意なのは「十分学習できるだけの情報が蓄積された事例」を再現すること。
  - 専門性が高い研究ソフトウェア開発には十分学習できるだけの事例がない。
  - AIにやらせるにしても、その方法論を学ぶのは研究現場にはまだ負担が大きい。

まとめにかえて

- ソフトウェアは現代の研究の重要なインフラストラクチャであり、ソフトウェア技術者は実験装置のオペレータや図書館司書に匹敵するインフラの担い手である。
- 研究者に不足している計算機に関する専門性を補い研究力を向上させる。
- ソフトウェア開発は費用対効果が高い（基本的に人件費のみ）。
- 優れたソフトウェアは共同研究のテーマや次世代育成のための教材になる。
（望遠鏡や加速器などの大型装置の開発と同等以上のインパクトがあり得る。）
- ソフトウェアは（公開すれば）機関や計算機、国を問わずどこでも利用可能。また、優れた設計のソフトウェアはハードウェアの世代を超えて利用可能。
→ 共同研究や国際協力のツールとして高い波及効果が期待できる。
- 研究に関わる人材に多様なキャリアパスを提供する。
（高い専門性を評価し、お互い尊敬できるような待遇が必要。）
- ソフトウェア開発に関わる人材を集約・育成することは、研究装置のコアファシリティ化の流れに合致する。

バックアップ

コード高速化

- コード高速化のノウハウは（知らなければ黒魔術だが）、わかってしまえば特別なスキルではなく、多くのコードに適用できる。
- 性能の限界を追求するには個別のコードに関する深い理解と非常に高いスキルが要求されるが、「そこそこ（例えば2倍）」の性能改善でも大きなメリットがある。
- 近年多いPythonはC/Fortran系に移植するだけで大幅な性能向上が期待できる。
- 研究現場においてはコード開発するフェイズと計算を実行して成果を出すフェイズが交互にあるため、常にコード開発/高速化の作業が発生するわけではない。
- このような計算機技術を持つ人材は研究現場には少なく、また研究者自身が行うにはやや（？）敷居が高い。
- 一部計算機センターではそのようなサポートがあるが、特定の計算機に紐付けになっている上に必ずしも技術力が高いわけではない。

⇒ 技術を有するソフトウェアエンジニアを数名雇用して、学内のコード開発・最適化を請け負う体制を構築すれば、研究者にとって非常に有用な支援になる。

新技術対応

- 近年では大型計算機の一部は電力多性能比に優れるGPUに移行している。この傾向は少なくとも今後数年は続く。GPUを含むアクセラレータでは従来のCPUとは異なるコード開発技術が必要になる。これは研究者には大きな負担：
  - CUDA/HIP等の専用言語
  - OpenMP/OpenACC等の指示文
  - Kokkos等のPerformance portability ソフトウェア
- 近年ではAI・機械学習・データ科学など新たな技法も急速に発展し、研究現場での応用も盛んに行われている。
  - 機械学習によるデータ解析の自動化
  - パラメータの補間や探索領域の最適化
  - Physics-Informed Neural-Network等による物理そのものへの応用
  - ビッグデータの解析や、データ同化技法による現実世界への応用

これらの新技術に関する知見を集積し研究者に提供するソフトウェアコンサルタント的な部署があれば、研究者にとって有用だろう。

手法を通じた学際研究

流体力学や画像処理などは理学工学通じて多くの分野で共通の課題であるにもかかわらず分野横断での連携は残念ながら限定的。

例えば流体力学では
- 共通点：格子法、粒子法、格子生成、並列化、データIO
- 相違点：必要な物理（圧縮/非圧縮・粘性/非粘性・化学反応・重力 etc.）

などの共通点もあり協力できるはずだが、現時点ではほぼ独立にコード開発が行われている。

ソフトウェアあるいは計算技術という手法に関する知見を集積し分野を超えて展開することで、「手法」「ソフトウェア」を軸とした新たな学際研究の橋渡しにもなる。

技術者だけでなく研究者も組織に参画することで、学内の共同研究を結びつけるハブとしての機能を担うことができる。