コード生成 AI によるプログラムの自動チューニングと サービス展開に向けて

片桐孝洋 [1)], 林俊一郎 [2)], 椋木大地 [1)], 星野哲也 [1)]

1) 名古屋大学 情報基盤センター
2) 名古屋大学 大学院情報学研究科

katagiri@cc.nagoya-u.ac.jp

Code generative AI to Auto-tuning of Programs and Towards its Service Deployment

Takahiro Katagiri [1)], Shun-ichiro Hyashi[2)], Daichi Mukunoki [1)], Tetsuya Hoshino [1)]

1) Information Technology Center, Nagoya Univ.
2) Graduate School of Informatics, Nagoya Univ.

概要

本報告では、大規模言語モデル(LLM)を活用したコード生成 AI による HPC コードの自動生成を目指す HPC-GENIE プロジェクトを紹介する。また急速に広まる Vibe Coding の概念に基づき、HPC-GENIE プロジェクトの開発コンセプトを実現したプロトタイプである VibeCodeHPC について紹介する。本研究では VibeCodeHPC の適用ケーススタディとして、行列-行列積の自動チューニング(AT)事例を取り上げる。さらに、共同利用・共同研究拠点が供する GPU コンピューティングのサービスとしてコード生成 AI 機能を提供する場合における検討を行う。

1 はじめに

近年、生成 AI (Generative AI)は自然言語処理や画像生成、音声合成、コード生成、といった多様な領域において著しい進展を遂げている。特に大規模言語モデル (Large Language Models, LLM) は膨大な事前学習データを用いて高度な言語生成能力を獲得し、多様な知識や文脈理解をした出力を実現している[1]。この発展は、Transformer の登場とスケーリング則[2]、加えて教師あり学習[3]などの技術に支えられている。さらに生成 AI は、GPT-4 や Claude、Gemini などのモデルなどで、質問応答、要約、翻訳、プログラムコードの補完、さらには複雑な意思決定支援などの応用が多数、試みられている。

以上の背景に加えて、コード生成 AI は高性能計算 (High Performance Computing, HPC) 分野においても、分野特有の高度知識が要求される並列化支援や、最新アーキテクチャを意識した性能チューニング支援に有望と考えられる。従来、人工知能分野で行われていたルールベースや、HPC 分野で行われていた DSL (Domain-Specific Language) では実現できない、柔軟なコード最適化の支援が期待できる。特に HPC 分野では、20 年来我が国で研究開発されている**ソフトウェア自動チューニング(Software Auto-tuning、AT)**[4]-[6]技術を、LLM による生成 AI を用いたプロンプトエンジニアリングと融合させる試みが開始されている。AT とコード生成 AI 技術を融合することで、全く新しい HPC 基盤の創成が期待される。

2 HPC-GENIE プロジェクト

2.1 概要

HPC-GENIE (High-Performance Computing with GEnerative Neural Intelligence for Execution) プロジェクト (https://www.hpc.itc.nagoya-u.ac.jp/menu/hpc_genie.html) は、名古屋大学情報基盤センター、および、大学院情報学研究科の関連者で 2025 年 4 月に発足したコード生成 AI を活用した HPC プログラム自動生成プロジェクトである。LLM を活用したプロンプトエンジニアリングと AT 技術を融合した AI 基盤により、チューニングやコード生成の自動化を達成する。このことで、

HPC ソフトウェア開発の生産性を劇的に高めることを目的としている。

従来の AT 技術では、主な技術基盤としては、テンプレート生成と探索手法 (遺伝的アルゴリズム/焼きなまし法/ベイズ最適化、など) を対象にしていた。これに対して HPC-GENIE プロジェクトでは、LLM を用いることが大きな違いである。また、探索手法では、従来の AT 技術ではヒューリスティクスと性能計測ベースで最適コード探索を行っていた。しかし HPC-GENIE プロジェクトでは、**反復プロンプトによる自己改善**により、最適コードの探索を目指す。

LLM の活用による欠点もある。最適化コードの正確性(精度)と再現性は、LLM ベースの方法では問題となる。従来の AT 技術では、プログラム実行を伴う計測ベースであり、コード生成に関しても基本的には決定的手法であり、高い再現性のあるコード生成が可能である。一方で LLM ベースのコード生成は、言語モデルの特性から確率的生成になる。そのため、生成されるコード自体にばらつきが生じる。この理由から、LLM ベースのコード最適化は再現性が高くなく、不安定である。加えてユーザが与えるプロンプトにより、大きく自動生成コードの品質が影響するという欠点がある。

2.2 設計思想

HPC-GENIE プロジェクトでは、以下の設計思想で LLM によるコード生成基盤の開発を行う。

1. HPC コード生成を実現するためのプロンプトエンジニアリング方式を開発する
2. 複数の LLM が利用できる CLI (Command Line Interface)の開発をする。また、AT 機能連携を行う。
3. HPC コード生成を実現するための RAG (Retrieval Augmented Generation) とファインチューニング機能の開発する
4. HPC コード生成を実現するためのローカル LLM を活用する
5. コード生成 AI による、精度保証、混合精度演算、説明可能 AI (XAI)、及び AT 機能を持つコードの自動生成を行う

表 1 は従来の LLM によるコード生成 AI 手法と HPE-GENIE プロジェクトによる狙いのまとめである。

特に 5 の、精度保証や混合精度演算のコード生成ができる機能は、現在のコード生成 AI では存在しない。かつ本機能は HPC や数値シミュレーション分野のコード特有の機能要求であるといえる。

表 1　従来の LLM によるコード生成 AI 手法と HPC-GENIE での狙い

項目	LLM による従来手法	HPC-GENIE の狙い
コード生成方法	汎用 LLM に手動プロンプト入力/回答利用	プロンプトエンジニアリング＋CLI ベースの複数 LLM 選択方式
AT 連携	CLI ツール (OpenTuner, GPTune 等) で手動接続	AT 統合型 (コード生成直後に AT 走査)
RAG 機能	一部 LLM で統合(LangChain など)	RAG 機構を明示的に内包する
精度保証/混合精度演算制御	演算精度は人手調整、混合精度演算は対応していない	精度保証/混合精度演算を明示的に統合する
説明可能 AI (XAI) 連携	ほぼ非対応 (black-box 出力)	XAI 要素(実行時間/演算精度/電力などの予測の妥当性検証)の統合を計画
ローカル LLM 対応	基本はクラウド LLM (グローバル LLM) 利(ChatGPT、Claude 等)	Swallow LLM 等のローカル LLM を含む(グローバル LLM も選択可)

現状のコード生成 AI は、性能に影響するパラメタに対する AT 機能は無く、かつ、AT 機能を有するプログラム生成を明示的に含んだシステムではない。コード生成 AI に AT 機能を明示的に統合する点が、HPC-GENIE プロジェクトの独創的な点である。

我々は、3 のローカル LLM の利用は、今後重要なテーマになると考えている。現状、商用 LLM (クラウド利用での LLM) の品質は、オープンソースで利用できるローカル LLM 品質より高い。一方、

機密性の高いコード開発では、セキュリティの都合から、いくら性能が高くても商用 LLM が利用できないケースは多い。このような場合において、品質の高いコード生成ができるフレームワークの開発が、今後ますます重要になるといえる。そのための品質改良の機能開発が、重要な HPC-GENIE プロジェクトのテーマである。

3 VibeCodeHPC の開発とケーススタディ

3.1 概要

我々は、2 章の設計方針を実現するプロトタイピングとして、VibeCodeHPC を開発している。

VibeCodeHPC は、Vibe Coding の概念に基づき、HPC ソフトウェアのチューニングやコード生成ができる AT 基盤である。ここで Vibe Coding とは、2025 年 7 月に Andrej Karpathy が提唱した成果物を開発する新しい方法論である。従来のプログラミングワークフローとは異なり Vibe Coding では、直接的なコード作成を最小限に抑え、ユーザが技術仕様よりも直感的な意図表現を優先する。会話型フローの中で「何かを見て、何かを言って、何かを実行する」ことを可能にするプログラム開発手法である。つまり、簡単に「プロトタイピング」ができるコード開発体系といえる。

3.2 基本機能

VibeCodeHPC は、 Multi Agentic Vibe Coding for HPC の基盤であり、2025 年 7 月 28 日に GitHub で公開[7]した。HPC コード最適化のための自動チューニングを行うマルチ AI エージェントシステムである。現在のバージョンでは、ClaudeCode4 の CLI 機能を利用している。Ver.1 のマルチ AI エージェントによる起動画面を図 1 に示す。

HPC-GENIE プロジェクトでの目的は、入力された HPC コード(Fortran 等)から、OpenMP、MPI、OpenACC、CUDA 等のコード生成を目指すものである。しかし、この用途に限らず、多様な利用法が考えられる。

VibeCodeHPC の特徴[7]は、以下である。

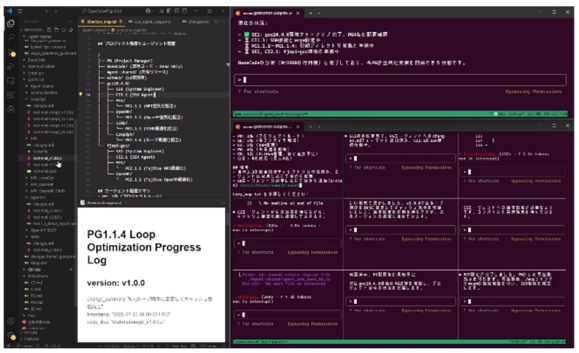

図 1　VibeCodeHPC の起動画面[7]

- 階層型マルチエージェントにより、プロジェクトマネージャ(PM) → システムエンジニア(SE)↔ プログラマ(PG)の AI エージェント割り当てにより、 企業的分業体制が構築可能
- プロジェクト地図: 組織をリアルタイムに視覚化する directory_pane_map を提供
- 進化的探索: ボトムアップ型の Flat 構造による効率的探索が可能
- 自動最適化: OpenMP、MPI、OpenACC、CUDA... 等の段階的並列化と技術融合が可能
- 予算管理: 計算資源の効率的配分と追跡が可能
- 統一ログ: ChangeLog.md による一元的な進捗管理が可能

現在の対応環境は、スパコン環境： スーパーコンピュータ「不老」や「富岳」等の HPC システムである。コンパイラは、Intel OneAPI、GCC、NVIDIA HPC SDK、等に対応している。

3.3 ケーススタディと予備性能評価

我々は、基本的な数値計算のカーネルコードをケーススタディとして、予備性能評価を行っている。実験対象は、性能チューニングの AT ジョブフローである。概要は以下である。

- 与えたコードの概要

行列-行列積の C コードを与える。単純な 3 重ループで実装されている。結果検証の機能付である。理論解と検証ルーチンが内包されている。

- コードチューニング課題

対象となる関数(MyMatMat)中の 3 重ループをループアンローリングによる高速化をせよという課題を与える。基本となる演算カーネルは、以下である。

```
for (i=0; i<n; i++) {
  for (j=0; j<n; j++) {
    for (k=0; k<n; k++) {
      C[i][j] += A[i][k] * B[k][j];
    }
  }
}
```

ここで、想定される VibeCodeHPC のジョブフローは以下である。

- 反復プロンプトのジョブフロー
  1. 手元の PC で LLM によるコード生成
  2. 生成コードをスパコンへ転送
     (例：スーパーコンピュータ「不老」TypeI サブシステム)
  3. 「不老」TypeI のログインノード上で make して実行可能コードを作成
  4. ジョブスケジューラへ pjsub コマンドによりジョブを投入し実行する
  5. 結果確認(実行速度の確認)
  6. LLM によるコード修正の指示
  7. 1 に戻る

以上のジョブフローが完全自動で実行される。実行デモとチューニング結果の提示は、当日発表で行う。

4 コード生成 AI サービスに向けて

我々は、HPC-GENIE プロジェクトにより、図 2 に示す「汎用コード生成 AI 環境」の構築を目指している。

図 2 では、ローカルサーバ上で、ローカル LLM 活用や RAG を経由してのグローバル LLM (商用 LLM)を活用した汎用のコード生成 AI 環境である。しかしながら、コード生成品質の向上には**高速推論**(マルチ AI エージェントの高速化)と、ユーザとの対話による知識を取得したうえでの**ファインチューニング**を行う必要がある。そのため、高速推論やファインチューニングの高速化のための最新 GPU の活用、および、安価な GPU クラスタの利用が必須と考えている。

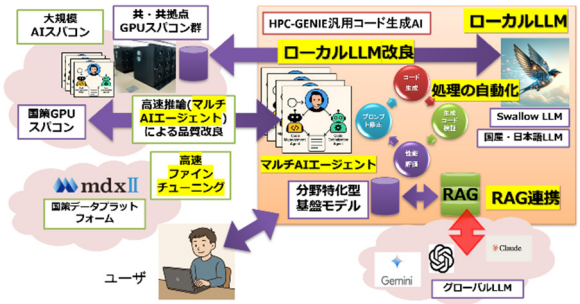

図 2　HPC-GENIE プロジェクトで目指す
汎用コード生成 AI 環境

以上の理由から、国策データプラットフォームである mdxII や、共同利用・共同研究拠点 (共・共拠点) のスパコン、もしくは、国策スパコンの「富岳 NEXT」の利用についても、システムと連携した実装の研究する必要があると考えている。

我々は特に、共・共拠点のスパコン利活用は、鍵となると考えている。その理由は、民間クラウドよりも安価なアカデミック用の課金体制があることによる。共・共拠点のスパコンによる、マルチ AI エージェント実行、および、ファインチューニングの実行ができる基盤を整備し、コード生成 AI サービスとして活用していく必要がある。

現在、LLM システム構築は、オープンソースソフトウェアを用いて行うことができる。また、多くのセンターの GPU スパコンで標準的に提供されている Singulality などのコンテナ環境を使えば、GPU 上でコード生成 AI の基本機能が実行できる。そのため、技術的には難しくないといえる。

今後、図 2 の汎用コード生成 AI 環境の機能の一部を担う、共・共拠点のスパコンのサービスについて、名古屋大学の次期スーパーコンピュータシステム「不老 NEXT」(仮称)の実利用を想定したサービス案を検討していく予定である。

5 おわりに

本報告では、大規模言語モデル(LLM)を活用したコード生成 AI の指数関数的な性能向上を考慮し、HPC コードの自動生成を目指す HPC-GENIE プロジェクトの開発方針について説明した。また、急速に広まる Vibe Coding の概念に基づき、HPC-GENIE プロジェクトの開発コンセプトを実現したプロトタイプである VibeCodeHPC について紹介した。

現在、我々は VibeCodeHPC の AI エージェント

機能開発とその予備性能評価を行っている。本稿では、ケーススタディとして行列-行列積の自動チューニング(AT)事例を取り上げた。予備評価中ではあるが、従来の AT システムでは達成困難と思われるコード生成や性能チューニングも、LLM によるコード生成 AI 環境で完全自動化が目指せる可能性がある。このようにコード生成 AI 技術は、目覚ましい発展があると認識をしている。

今後、特にローカル LLM の活用技術が課題となると予想される。その際には、共同利用・共同研究拠点が供する GPU コンピューティングのサービスの一環として、マルチ AI エージェントによる高速推論と高速ファインチューニングのサービスが重要になることを既に説明した。今後、センターの実サービスとして提供する場合の課題の検討を進めていく予定である。

LLM によるコード生成 AI の HPC コード生成の研究では、我々は既に数値計算ライブラリ BLAS の自動生成の研究[8]を行っている。これらの知見を融合させる VibeCodeHPC の機能開発は、今後の重要な課題であるといえよう。

また現在、BLAS などの基本計算ルーチン以外について、実用的な数値計算の演算カーネルコードに対してコード生成 AI の適用評価を試みている。この評価を通して、LLM によるコード生成の適用限界の調査と問題提起、および、解決手法の提案が必須であり、今後の最重要課題である。

謝辞

本研究は、学際大規模情報基盤共同利用・共同研究拠点（JHPCN）、および、革新的ハイパフォーマンス・コンピューティング・インフラ（HPCI）の支援による(課題番号: jh250015)。本研究は JSPS 科研費 JP23K11126，JP24K02945 の助成を受けたものです。

参考文献

[1] OpenAI et al., GPT-4 Technical Report, arXiv:2303.08774, 2023.
[2] A. Vaswani et al., Attention Is All You Need, Proc. of NIPS2017, 2017.
[3] J. Devlin et al., BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, Proc. of NAACL-HLT2019, pp.4171–4186, 2019.
[4] T. Katagiri, K. Kise, H. Honda, T. Yuba, FIBER: A generalized framework for auto-tuning software, Proceedings of International Symposium on High Performance Computing (ISHPC)2003, pp.146-159, (2003).
[5] T. Katagiri, K. Kise, H. Honda, T. Yuba, ABCLibScript: A directive to support specification of an auto-tuning facility for numerical software, Parallel Computing, Vol. 32, No. 1, pp. 92-112, (2006).
[6] T. Katagiri, D. Takahashi, Japanese autotuning research: Autotuning languages and FFT, Proceedings of the IEEE, Vol. 106, No. 11, pp. 2056-2067, (2018)
[7] VibeCodeHPC の GitHub: https://github.com/Katagiri-Hoshino-Lab/VibeCodeHPC-jp
[8] D. Mukunoki, S. Hayashi, T. Hoshino, T. Katagiri, Performance Evaluation of General Purpose Large Language Models for Basic Linear Algebra Subprograms Code Generation, arXiv:2507.04697 [cs.LG], July 2025. A preprint.