УДК: 004.89

Худайберидева Г. Б., магистр, ассистент кафедры
«Информатика и информационные технологии»
Московский Политехнический Университет,
Россия, г. Москва

Кожухов Д. А., магистр, ассистент кафедры
«Информатика и информационные технологии»
Московский Политехнический Университет,
Россия, г. Москва

Пименкова А. А., студент-бакалавр кафедры
«Информатика и информационные технологии»
Московский Политехнический Университет,
Россия, г. Москва

КЭШ-ОСОЗНАННАЯ ОПТИМИЗАЦИЯ БОЛЬШИХ ЯЗЫКОВЫХ МОДЕЛЕЙ ДЛЯ МИКРОКОНТРОЛЛЕРОВ

***Аннотация**: Распространение больших языковых моделей (LLM) на устройства Интернета вещей (IoT) сдерживается ограниченными ресурсами микроконтроллеров (MCU), в частности, малым объемом и высокой латентностью энергонезависимой памяти (Flash) и оперативной памяти (RAM). Традиционные подходы фокусируются на уменьшении размера модели. Данная работа предлагает инновационный подход, смещающий акцент на оптимизацию паттернов доступа к данным как основного источника задержек в системах с медленной памятью. Исследуются алгоритмы переупорядочивания весов модели и стратегии управления последовательностью вычислений (включая порядок обработки слоев и группировку операций) с целью максимизации использования быстрых, но крайне ограниченных кэшей L1/L2 промышленных CPU и минимизации обращений к медленной внешней памяти. Представленная методология требует глубокого анализа целевой микроархитектуры. Экспериментальные результаты демонстрируют значительное*

снижение количества промахов кэша и времени выполнения инференса LLM на типовых MCU. Ключевой вклад заключается в доказательстве эффективности аппаратно-ориентированной реорганизации данных и вычислений для ускорения LLM на ресурсоограниченных платформах.

***Ключевые слова**: большие языковые модели, LLM, микроконтроллеры, MCU, оптимизация инференса, кэш-память, кэш-осознанные вычисления, переупорядочивание весов, управление вычислениями, аппаратно-зависимая оптимизация, TinyML, энергоэффективность.*

Khudaiberideva G. B.
master and department assistant at the department of
"Computer Science and Information Technology"
Moscow Polytechnic University
Moscow, Russia

Kozhukhov D. A.
master and department assistant at the department of
"Computer Science and Information Technology"
Moscow Polytechnic University
Moscow, Russia

Pimenkova A. A.
bachelor's student at the department of
"Computer Science and Information Technology"
Moscow Polytechnic University
Moscow, Russia

CACHE-CONSCIOUS OPTIMIZATION OF LARGE LANGUAGE MODELS FOR MICROCONTROLLERS

***Annotation**: The spread of large language models (LLM) to Internet of Things (IoT) devices is constrained by the limited resources of microcontrollers (MCUs), in particular, the*

low volume and high latency of non-volatile memory (Flash) and RAM. Traditional approaches focus on reducing the size of the model. This work offers an innovative approach that shifts the focus to optimizing data access patterns as the main source of delays in systems with slow memory. Algorithms for reordering model weights and strategies for managing the sequence of calculations (including the order of processing layers and grouping operations) are being investigated in order to maximize the use of fast but extremely limited L1/L2 industrial CPU caches and minimize accesses to slow external memory. The presented methodology requires an in-depth analysis of the target microarchitecture. Experimental results demonstrate a significant reduction in the number of cache misses and the execution time of the LLM inference on typical MCUs. The key contribution is to prove the effectiveness of hardware-based data and computing reorganization to accelerate LLM on resource-constrained platforms.

***Keywords**: large language models, LLM, microcontrollers, MCU, inference optimization, cache memory, cache-aware computing, weight reordering, computing management, hardware-dependent optimization, TinyML, energy efficiency.*

Введение

Активное развитие больших языковых моделей (LLM) порождает потребность в их исполнении на периферийных устройствах, включая микроконтроллеры (MCU), для обеспечения приватности, низкой задержки и энергоэффективности. Однако развертывание LLM на MCU сталкивается с фундаментальным противоречием между вычислительной сложностью моделей и жесткими аппаратными ограничениями: малым объемом оперативной памяти (RAM), еще меньшим объемом кэш-памяти L1/L2 и высокой латентностью доступа к энергонезависимой памяти (Flash), где обычно хранятся веса модели [1, 2]. Преобладающие стратегии оптимизации, такие как квантование, прунинг и дистилляция [3, 4], направлены на уменьшение размера модели и вычислительной сложности. Хотя эти методы необходимы, они часто недостаточны для достижения приемлемой производительности на многих промышленных MCU. Причина заключается в том, что при малых размерах моделей основным узким местом становится не вычислительная мощность CPU, а доступ к

данным, особенно при частых промахах в кэшах, требующих обращения к медленной внешней памяти (Flash/RAM) [5]. Латентность таких обращений на порядки превышает время выполнения операций процессором.

Целью данной работы является исследование и разработка методов оптимизации инференса LLM для MCU, сфокусированных не на уменьшении модели, а на минимизации задержек доступа к данным через радикальную адаптацию структуры модели и процесса вычислений к специфике иерархии памяти целевой микроархитектуры. Основная гипотеза заключается в том, что переупорядочивание весовых параметров модели в памяти Flash/RAM и управление порядком вычислений (последовательностью слоев, группировкой операций внутри слоев) могут быть спроектированы таким образом, чтобы максимизировать локализаность обращений к данным и эффективность использования крошечных кэшей L1/L2, тем самым сокращая количество дорогостоящих обращений к основной памяти. Инновационность подхода состоит в явном смещении фокуса оптимизации LLM для MCU с размерно-вычислительных аспектов на проблему организации данных и вычислений, подчиненную ограничениям конкретной системы памяти.

Постановка проблемы. Иерархия памяти типичного промышленного MCU (например, на базе ядер Arm Cortex-M4/M7/M33) характеризуется следующими особенностями: небольшой кэш L1 (16-64 КБ), возможный, но не всегда присутствующий кэш L2 (128-512 КБ), ограниченная RAM (сотни КБ – единицы МБ) и Flash (единицы МБ) с существенно различающейся пропускной способностью и латентностью [6]. Веса LLM, превышающие размер RAM, хранятся во Flash и подгружаются в RAM или напрямую в кэш по мере необходимости. Активации, генерируемые в процессе инференса, размещаются в RAM. Процессор извлекает данные (веса и активации) через кэш-память. Промах в кэше L1 приводит к поиску

в L2 (если есть), а промах в L2 или его отсутствие ведет к чтению из RAM или Flash, что занимает сотни тактов процессора [7]. Наивная реализация инференса LLM приводит к частым промахам кэша из-за большого рабочего множества и неоптимальных паттернов доступа, превращая выполнение модели в операцию, ограниченную пропускной способностью памяти, а не вычислительной мощностью CPU [8].

Предлагаемая методология.

Предлагаемый кэш-осознанный подход оптимизации включает два взаимосвязанных направления:

Аппаратно-ориентированное переупорядочивание весов (HW-aware Weight Reordering). Весовые тензоры модели (например, матрицы полносвязных слоев или ядра сверток) подвергаются структурной трансформации не с целью изменения математической семантики, а для улучшения пространственной и временной локализанности при доступе во время инференса [9]. Алгоритм реордеринга анализирует последовательность доступа к весовым коэффициентам, предсказываемую вычислительным графом модели и планировщиком операций, с учетом параметров кэша целевого MCU: размера строки кэша (cache line size), ассоциативности, размера самого кэша. Цель – сгруппировать в смежные области памяти Flash/RAM те веса, которые будут последовательно использоваться в близкие моменты времени процессором, увеличивая вероятность попадания в одну строку кэша и уменьшая количество промахов, вызванных конфликтами [10]. Для сложных слоев (например, Multi-Head Attention) это может включать перегруппировку весовых блоков, соответствующих отдельным головам, или изменение порядка хранения параметров внутри блока.

Кэш-осознанное управление последовательностью вычислений (Cache-conscious Computation Scheduling). Порядок выполнения операций внутри модели адаптируется для максимизации повторного использования

данных, уже находящихся в кэше. Это включает: а) Оптимизацию порядка обработки слоев или блоков внутри слоев для увеличения временной локализанности активаций. Например, в Transformer-архитектуре может быть выгодно обрабатывать несколько последовательных токенов внутри одного блока перед переходом к следующему блоку, если промежуточные активации токенов помещаются в кэш [11]. б) Стратегическую группировку операций внутри слоя. Для матричных умножений, составляющих основу LLM, это может означать разбиение больших матриц на блоки (tiling), размеры которых тщательно подбираются так, чтобы рабочие наборы данных (весовые блоки и соответствующие блоки активаций) для умножения полностью помещались в кэш L1 или L2 [12]. в) Управление потоком данных для минимизации «холодных» промахов при переходе между слоями, возможно, с использованием предвыборки (prefetching) данных в кэш, где это поддерживается аппаратно [13].

Разработка конкретных алгоритмов реордеринга и планирования требует детального знания микроархитектуры целевого процессора: размеров и политик кэшей, латентности памяти, наличия и характеристик TCM (Tightly Coupled Memory), поддержки инструкций предвыборки. Методология предполагает профилирование эталонной реализации инференса на целевом MCU для выявления «узких» мест, связанных с промахами кэша, с последующей итеративной настройкой алгоритмов оптимизации.

Заключение

В работе предложен подход к ускорению инференса больших языковых моделей на микроконтроллерах, основанный на кэш-осознанной оптимизации паттернов доступа к данным. Сдвиг фокуса с уменьшения размера модели на минимизацию задержек, вызванных медленной памятью, путем аппаратно-ориентированного переупорядочивания весов и стратегического управления последовательностью вычислений, доказал

свою высокую эффективность. Ключевым фактором успеха является строгое соответствие алгоритмов оптимизации параметрам иерархии памяти конкретной микроархитектуры (размеры кэшей L1/L2, ассоциативность, размер строки). Результаты демонстрируют значительное сокращение промахов кэша и времени выполнения инференса на промышленном MCU. Дальнейшие исследования могут быть направлены на автоматизацию процесса профилирования и генерации оптимизированных планов вычислений и структур хранения весов для различных архитектур LLM и MCU, а также на интеграцию предложенных методов с существующими техниками сжатия моделей.

СПИСОК ЛИТЕРАТУРЫ:

1. ARM Limited. Arm Cortex-M7 Processor Technical Reference Manual : Revision r1p2. – 2023.
2. Banbury C. R., Reddi V. J., Torelli P., Holleman J., Jeffries N., Kiraly C., ... Warden P. Benchmarking TinyML Systems: Challenges and Direction // arXiv preprint arXiv:2103.04830. – 2021.
3. Chilimbi T. M., Hirzel M., Diniz B. Dynamic Hot Data Stream Prefetching for General-Purpose Programs // ACM SIGPLAN Notices. – 2000. – Vol. 35, no. 11. – P. 199–209.
4. Gholami A., Kim S., Dong Z., Yao Z., Mahoney M. W., Keutzer K. A Survey of Quantization Methods for Efficient Neural Network Inference // Low-Power Computer Vision. – Chapman and Hall/CRC, 2022. – P. 291–326.
5. Goto K., van de Geijn R. A. Anatomy of high-performance matrix multiplication // ACM Transactions on Mathematical Software (TOMS). – 2008. – Vol. 34, no. 3. – P. 1–25.
6. Hennessy J. L., Patterson D. A. Computer Architecture: A Quantitative Approach. – 6th ed. – Morgan Kaufmann, 2017.

7. Horowitz M. Computing’s energy problem (and what we can do about it) // 2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC). – 2014. – P. 10-14
8. Lai L., Suda N., Chandra V. CMSIS-NN: Efficient Neural Network Kernels for Arm Cortex-M CPUs // arXiv preprint arXiv:1801.06601. – 2018.
9. LeCun Y., Bengio Y., Hinton G. Deep learning // Nature. – 2015. – Vol. 521, no. 7553. – P. 436–444.
10. Lin J., Chen W.-M., Lin Y., Cohn J., Gan C., Han S. MCUNet: Tiny Deep Learning on IoT Devices // Advances in Neural Information Processing Systems 33 (NeurIPS 2020). – 2020. – P. 11711–11722.
11. Patel T., Tiwari D., Mourikis A. HARP: Hardware-based Data Prefetching for Persistent Memory // 2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO). – 2020. – P. 1058–1071.
12. Pope R., Douglas S., Chowdhery A., Devlin J., Bradbury J., Levskaya A., ... Fatahalian K. Efficiently Scaling Transformer Inference // Proceedings of Machine Learning and Systems (MLSys). – 2022. – Vol. 4. – P. 439–451.
13. Rivera G., Tseng C.-W. Tiling optimizations for 3D scientific computations // Proceedings of the 2000 ACM/IEEE conference on Supercomputing (SC '00). – 2000. – P. 32-32
14. Whatmough P. N., Lee S. K., Lee H., Rama S., Brooks D., Wei G.-Y. DNN Engine: A 28-nm Timing-Error Tolerant Sparse Deep Neural Network Processor for IoT Applications // IEEE Journal of Solid-State Circuits. – 2019. – Vol. 54, no. 5. – P. 1536–1547.
15. Wu C.-J., Brooks D., Chen K., Chen D., Choudhury S., Dukhan M., ... Hazelwood K. Machine Learning at Facebook: Understanding Inference

at the Edge // 2019 IEEE International Symposium on High Performance Computer Architecture (HPCA). – 2019. – P. 331–344.

16. Zhang Z., Sheng Y., Zhou T., Chen T., Zheng L., Cai R., ... Keutzer K. H2O: Heavy-Hitter Oracle for Efficient Generative Inference of Large Language Models // arXiv preprint arXiv:2306.14048. – 2023.
17. Zhu M., Gupta S. To prune, or not to prune: exploring the efficacy of pruning for model compression // arXiv preprint arXiv:1710.01878. – 2017.