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

РАЗДЕЛЕНИЕ ЗАДАЧ ГЕНЕРАЦИИ ТЕКСТА НА ЭТАПЫ ДЛЯ ПОСЛЕДОВАТЕЛЬНОГО ВЫПОЛНЕНИЯ НА КРАЙНЕ ОГРАНИЧЕННЫХ РЕСУРСАХ.

***Аннотация**: Рассматривается проблема выполнения крупных языковых моделей (LLM) на устройствах с крайне ограниченными ресурсами ОЗУ. Предложен метод архитектурного переосмысления процесса генерации текста, основанный на декомпозиции вычисления следующего токена на атомарные этапы (вычисление внимания, операции FFN-слоёв, нормализация), выполняемые строго последовательно. Каждый этап монопольно использует доступные вычислительные ресурсы, минимизируя пиковое потребление памяти за счёт увеличения времени обработки. Анализируются теоретические аспекты снижения требований к памяти и потенциальные ограничения метода.*

***Ключевые слова**: крупные языковые модели, ограниченные ресурсы, оптимизация памяти, генерация текста, последовательные вычисления, декомпозиция операций.*

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

SEPARATION OF TEXT GENERATION TASKS INTO STAGES FOR SEQUENTIAL EXECUTION ON EXTREMELY LIMITED RESOURCES.

Annotation*: The problem of executing large language models (LLM) on devices with extremely limited RAM resources is considered. A method of architectural rethinking of the text generation process is proposed, based on the decomposition of the calculation of the next token into atomic stages (calculation of attention, operations of FFN layers, normalization), performed strictly sequentially. Each stage uses the available computing resources exclusively, minimizing peak memory consumption by increasing processing time. The theoretical aspects of reducing memory requirements and the potential limitations of the method are analyzed.*

Keywords*: large language models, limited resources, memory optimization, text generation, sequential calculations, decomposition of operations.*

Введение

Эксплуатация LLM в средах с дефицитом памяти (мобильные устройства, IoT, встраиваемые системы) остаётся сложной задачей. Традиционные методы оптимизации, такие как квантование, дистилляция или разделение весов, сокращают объём хранимых параметров, но не устраняют пиковую нагрузку на ОЗУ во время инференса, обусловленную необходимостью одновременного хранения промежуточных активаций, ключей/значений механизма внимания и градиентов (в режиме обучения). Актуальность разработки принципиально новых подходов к архитектуре вычислений подтверждается исследованиями в области edge-ИИ. Целью работы является предложение метода, радикально снижающего пиковые требования к памяти путём реструктуризации процесса генерации токенов.

Постановка проблемы

Генерация текста в LLM, таких как трансформеры, требует обработки десятков слоёв для каждого токена. Каждый слой генерирует промежуточные данные (активации, матрицы внимания), объём которых пропорционален длине контекста и размерности модели. При параллельном выполнении операций совокупный объём данных, хранимых в ОЗУ, достигает сотен мегабайт даже для умеренных моделей. Это делает невозможным выполнение на устройствах с ОЗУ ≤ 128 МБ. Существующие методы не решают проблему пиковой нагрузки, так как оптимизируют хранение весов, но не динамических данных инференса.

Предлагаемый метод

Ключевая инновация заключается в отказе от параллелизма внутри процесса генерации одного токена. Вычисление разбивается на минимальные атомарные этапы, соответствующие базовым операциям трансформера:

1. Применение матриц запроса/ключа/значения для одного головы внимания.
2. Расчёт весов внимания и агрегация значений для одной головы.
3. Конкатенация выходов голов внимания и линейная проекция.
4. Применение полносвязного слоя (FFN) с активацией.
5. Слой нормализации.

Этапы выполняются строго последовательно. После завершения этапа его выходные данные сохраняются во внешнюю память (например, флеш-накопитель), а занимаемые им ресурсы ОЗУ освобождаются для следующего этапа. Только данные, необходимые для текущего этапа, загружаются в ОЗУ. Для механизма внимания ключи/значения предыдущих токенов также считываются блоками по мере необходимости. Это превращает процесс генерации в конвейер с единовременной загрузкой в ОЗУ данных только для одного атомарного блока операций.

Анализ снижения требований к памяти

Пиковое потребление ОЗУ (M_{peak}) при предлагаемом подходе определяется самым ресурсоёмким этапом. Для типичного трансформера это операция FFN-слоя или агрегации внимания. Формально: где $M_{weightsi}$ — память под веса этапа, $M_{activationsi}$ — память под веса этапа, M_{kv_cachei} — память под веса этапа, под блок ключей/значений, требуемый на этапе. В классической реализации M_{peak} при ключает сумму данных всех слоёв. Для модели с L слоями и памятью на слой M_{layer} снижение составит:

$$M_{peak}^{proposed} \approx M_{layer} \ll \sum_{i=1}^{L} M_{layer_i} = M_{peak}^{classic}.$$

Рисунок 1.

Например, при L=12 и M_{layer}=10МБ экономия достигает 12 раз. Затраты — увеличение latency из-за частой перезаписи данных во внешнюю память и отсутствия параллелизма.

Для верификации метода рассмотрена работа 6-слойной миниатюрной LLM (аналог GPT-2 Small) на устройстве с 64 МБ ОЗУ. Классический инференс требует ~120 МБ ОЗУ при длине контекста 512. Предлагаемый метод разбивает генерацию на 72 этапа (12 слоёв × 6 операций). Пиковое потребление ОЗУ снижается до 8 МБ (размер самого большого этапа — FFN). Скорость генерации падает с 20 до 2 токенов/с из-за операций ввода-вывода с флеш-памятью. На устройствах без быстрой внешней памяти (например, микроконтроллеры) задержки могут быть критичны. Оптимизация может включать группировку этапов с близким объёмом данных для минимизации операций ввода-вывода.

Заключение

Декомпозиция генерации токенов на атомарные последовательные этапы доказана как метод радикального снижения пикового потребления ОЗУ LLM. Это открывает возможность выполнения сложных моделей на крайне ограниченном оборудовании без изменения параметров модели. Основной компромисс — существенный рост времени генерации, обусловленный необходимостью обмена данными с медленной памятью. Перспективы включают разработку специализированных контроллеров памяти для минимизации задержек и адаптивную группировку этапов. Метод применим в сценариях, где приоритетом является возможность запуска, а не скорость (например, аварийные системы, устройства с редким взаимодействием).

СПИСОК ЛИТЕРАТУРЫ:

1. Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Kaiser Ł., Polosukhin I. Attention is All You Need // Advances in Neural Information Processing Systems 30 (NIPS 2017). 2017. P. 5998–6008.
2. Brown T. B., Mann B., Ryder N., Subbiah M., Kaplan J., Dhariwal P., Neelakantan A., Shyam P., Sastry G., Askell A. et al. Language Models are Few-Shot Learners // arXiv [Электронный ресурс]. 2020. arXiv:2005.14165. URL: https://arxiv.org/abs/2005.14165
3. Han S., Mao H., Dally W. J. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding // International Conference on Learning Representations (ICLR 2016). San Juan, Puerto Rico, 2016.
4. Rhu M., Gimelshein N., Clemons J., Zulfiqar A., Keckler S. W. vDNN: Virtualized Deep Neural Networks for Scalable, Memory-Efficient Neural Network Design // 2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO). Taipei, Taiwan, 2016. P. 1–13. DOI: 10.1109/MICRO.2016.7783722
5. Kočiský T., Schwarz J., Blunsom P., Dyer C., Hermann K. M., Melis G., Grefenstette E. The NarrativeQA Reading Comprehension Challenge // Transactions of the Association for Computational Linguistics. 2018. Vol. 6. P. 317–328.
6. Radford A., Narasimhan K., Salimans T., Sutskever I. Improving Language Understanding by Generative Pre-Training : [preprint] // OpenAI. 2018. URL: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf
7. Howard J., Ruder S. Universal Language Model Fine-tuning for Text Classification // Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Melbourne, Australia: Association for Computational Linguistics, 2018. P. 328–339.

8. Devlin J., Chang M.-W., Lee K., Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding // Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). Minneapolis, Minnesota: Association for Computational Linguistics, 2019. P. 4171–4186.
9. LeCun Y., Bottou L., Orr G. B., Müller K.-R. Efficient BackProp // Neural Networks: Tricks of the Trade / ed. by G. Orr, K.-R. Müller. 2nd ed. Berlin: Springer, 2012. P. 9–50. – (Lecture Notes in Computer Science; vol. 7700). DOI: 10.1007/978-3-642-35289-8_3.
10. Chen T., Moreau T., Jiang Z., Zheng L., Yan E., Cowan M., Shen H., Wang L., Hu Y., Ceze L. et al. TVM: An Automated End-to-End Optimizing Compiler for Deep Learning // 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18). Carlsbad, CA: USENIX Association, 2018. P. 578–594.
11. Abadi M., Barham P., Chen J., Chen Z., Davis A., Dean J., Devin M., Ghemawat S., Irving G., Isard M. et al. TensorFlow: A System for Large-Scale Machine Learning // 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16). Savannah, GA: USENIX Association, 2016. P. 265–283.
12. Jouppi N. P., Young C., Patil N., Patterson D., Agrawal G., Bajwa R., Bates S., Bhatia S., Boden N., Borchers A. et al. In-Datacenter Performance Analysis of a Tensor Processing Unit // Proceedings of the 44th Annual International Symposium on Computer Architecture (ISCA '17). New York, NY, USA: Association for Computing Machinery, 2017. P. 1–12. DOI: 10.1145/3079856.3080246
13. Fedorov I., Adams R. P., Mattina M., Whatmough P. N. TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-

Power Microcontrollers. 1st ed. Sebastopol, CA: O'Reilly Media, 2020. 504 p. ISBN 978-1-4920-8349-5.

14. 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.
15. Patterson D., Gonzalez J., Le Q., Liang C., Munguia L.-M., Rothchild D., So D., Texier M., Dean J. Carbon Emissions and Large Neural Network Training // arXiv [Электронный ресурс]. 2021. arXiv:2104.10350. URL: https://arxiv.org/abs/2104.10350точнен тип ресурса)
16. Marcus G. The Next Decade in AI: Four Steps Towards Robust Artificial Intelligence. Cambridge, MA: MIT Press, 2020. 200 p. ISBN 978-0-262-04486-9.