FLASHATTENTION

Васильевых Павел, БПМИ213

ПЛАН

1. Стандартный Attention.
2. FlashAttention.
3. FlashAttention 2.

СТАНДАРТНЫЙ ATTENTION

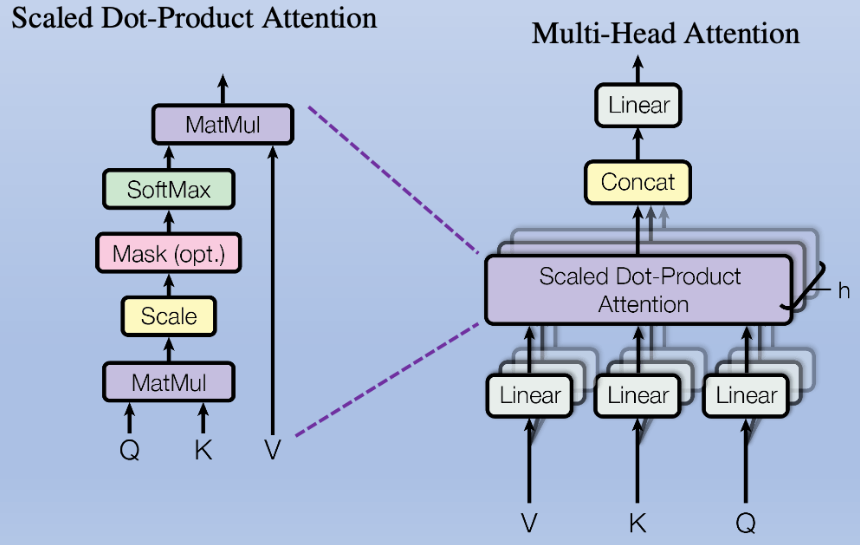
Algorithm 0 Standard Attention Implementation

Require: Matrices $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{N \times d}$ in HBM.

1: Load $\mathbf{Q}, \mathbf{K}$ by blocks from HBM, compute $\mathbf{S} = \mathbf{Q}\mathbf{K}^\top$, write $\mathbf{S}$ to HBM.
2: Read $\mathbf{S}$ from HBM, compute $\mathbf{P} = \mathrm{softmax}(\mathbf{S})$, write $\mathbf{P}$ to HBM.
3: Load $\mathbf{P}$ and $\mathbf{V}$ by blocks from HBM, compute $\mathbf{O} = \mathbf{P}\mathbf{V}$, write $\mathbf{O}$ to HBM.
4: Return $\mathbf{O}$.

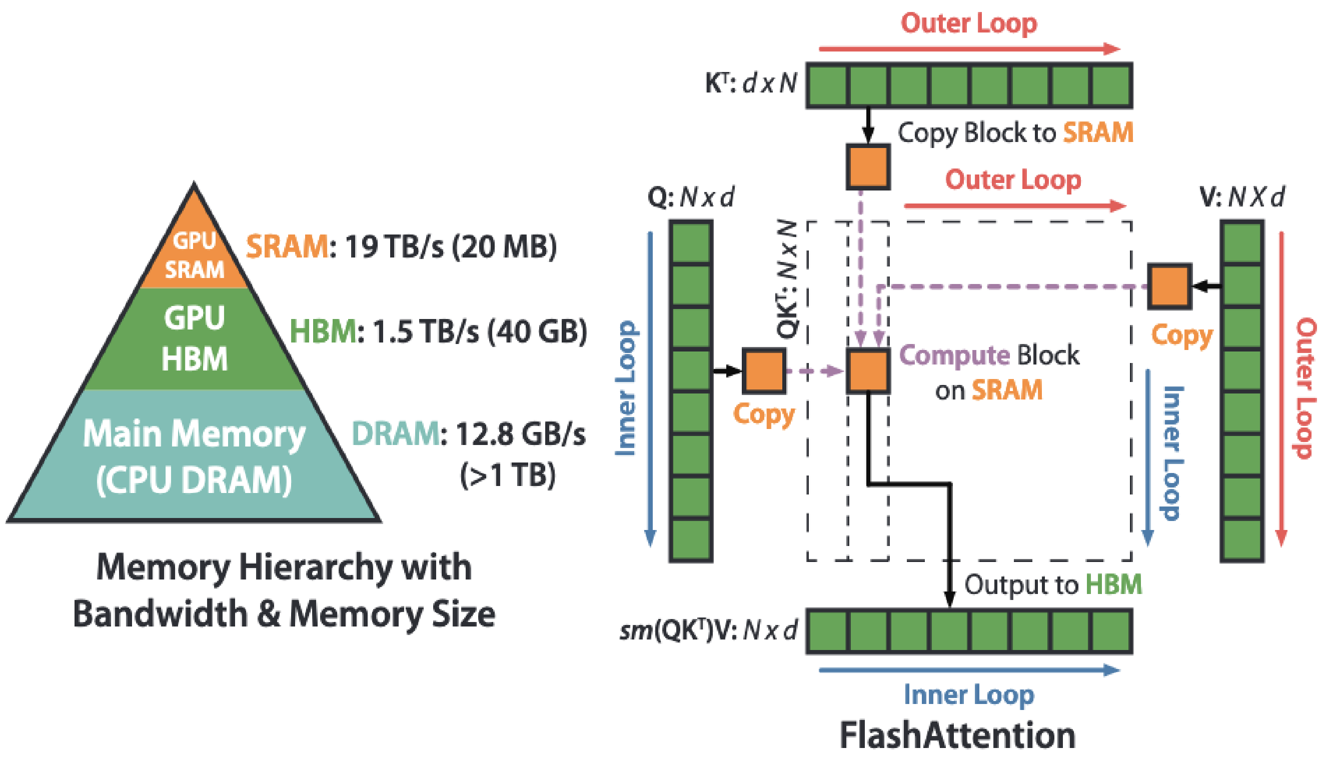
Количество обращений к медленной памяти равно

$$O(N^2 + Nd).$$

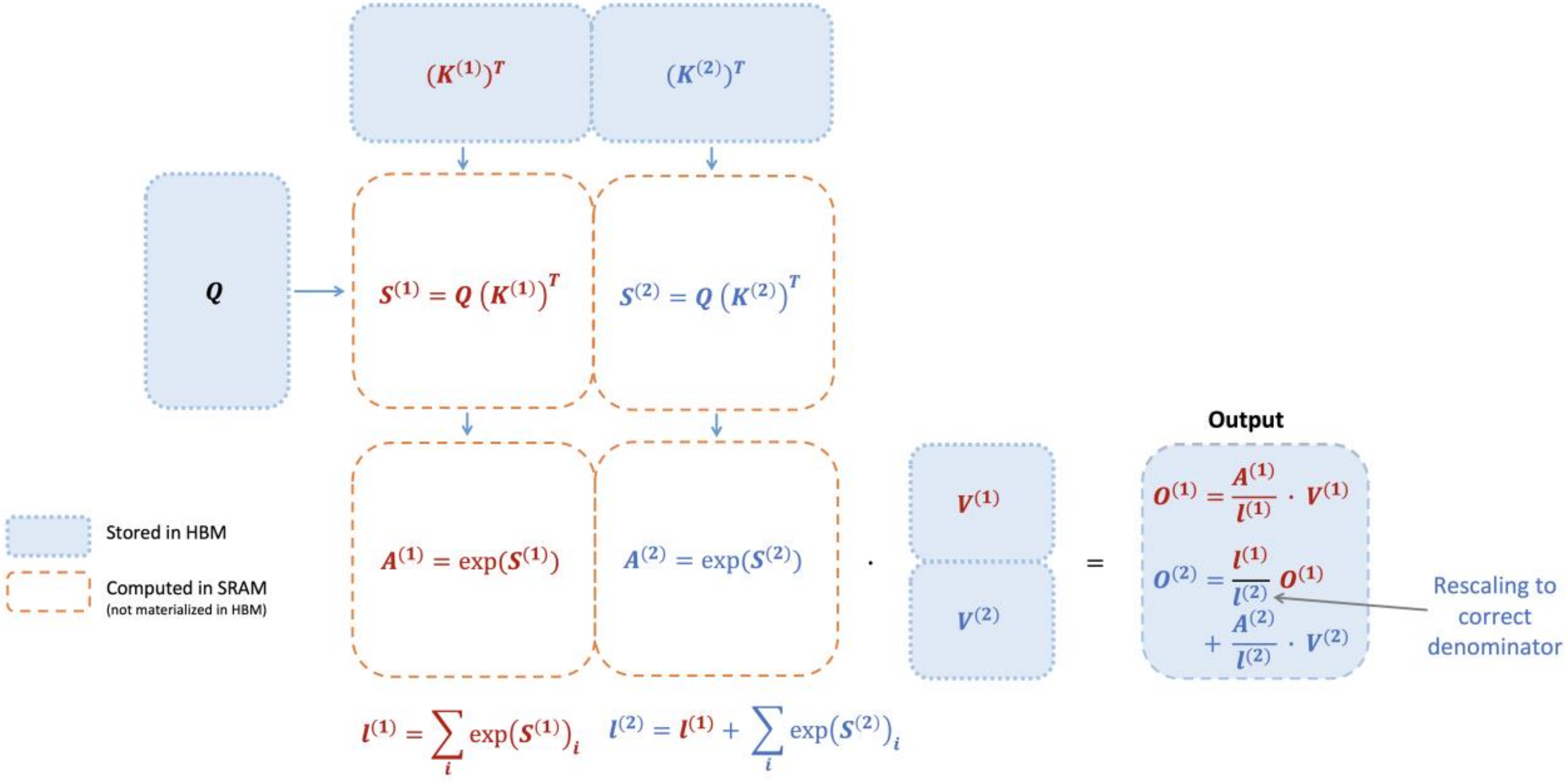


ПОЧЕМУ FLASHATTENTION?

Авторы заметили, что на время работы значительно влияют операции с памятью. Главная идея — уменьшить количество операций копирования между медленной и быстрой памятью.



УЛУЧШЕННЫЙ АЛГОРИТМ



УЛУЧШЕННЫЙ АЛГОРИТМ

Количество обращений к медленной памяти равно $O(N^2 d^2 M^{-1})$.

С помощью Block-Sparse FlashAttention можно сделать ещё быстрее.

Algorithm 1 FlashAttention

Require: Matrices $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{N\times d}$ in HBM, on-chip SRAM of size M.

Set block sizes $B_r = \left\lceil \frac{M}{4d} \right\rceil, B_c = \min\left(\left\lceil \frac{M}{4d} \right\rceil, d\right)$.

Initialize $\mathbf{O} = (0)_{N\times d} \in \mathbb{R}^{N\times d}, \ell = (0)_N \in \mathbb{R}^N, m = (-\infty)_N \in \mathbb{R}^N$ in HBM.

Divide $\mathbf{Q}$ into $T_r = \left\lceil \frac{N}{B_r} \right\rceil$ blocks $\mathbf{Q}_1, \dots, \mathbf{Q}_{T_r}$ of size $B_r \times d$ each, and divide $\mathbf{K}, \mathbf{V}$ in to $T_c = \left\lceil \frac{N}{B_c} \right\rceil$ blocks $\mathbf{K}_1, \dots, \mathbf{K}_{T_c}$ and $\mathbf{V}_1, \dots, \mathbf{V}_{T_c}$, of size $B_c \times d$ each.

Divide $\mathbf{O}$ into T_r blocks $\mathbf{O}_i, \dots, \mathbf{O}_{T_r}$ of size $B_r \times d$ each, divide ℓ into T_r blocks $\ell_i, \dots, \ell_{T_r}$ of size B_r each, divide m into T_r blocks $m_1, \dots, m_{T_r}$ of size B_r each.

for $1 \le j \le T_c$ **do**

  Load $\mathbf{K}_j, \mathbf{V}_j$ from HBM to on-chip SRAM.

  for $1 \le i \le T_r$ **do**

    Load $\mathbf{Q}_i, \mathbf{O}_i, \ell_i, m_i$ from HBM to on-chip SRAM.

    On chip, compute $\mathbf{S}_{ij} = \mathbf{Q}_i \mathbf{K}_j^T \in \mathbb{R}^{B_r \times B_c}$.

    On chip, compute $\tilde{m}_{ij} = \mathrm{rowmax}(\mathbf{S}_{ij}) \in \mathbb{R}^{B_r}$, $\tilde{\mathbf{P}}_{ij} = \exp(\mathbf{S}_{ij} - \tilde{m}_{ij}) \in \mathbb{R}^{B_r \times B_c}$ (pointwise), $\tilde{\ell}_{ij} = \mathrm{rowsum}(\tilde{\mathbf{P}}_{ij}) \in \mathbb{R}^{B_r}$.

    On chip, compute $m_i^{\mathrm{new}} = \max(m_i, \tilde{m}_{ij}) \in \mathbb{R}^{B_r}$, $\ell_i^{\mathrm{new}} = e^{m_i - m_i^{\mathrm{new}}} \ell_i + e^{\tilde{m}_{ij} - m_i^{\mathrm{new}}} \tilde{\ell}_{ij} \in \mathbb{R}^{B_r}$.

    Write $\mathbf{O}_i \leftarrow \mathrm{diag}(\ell_i^{\mathrm{new}})^{-1} (\mathrm{diag}(\ell_i) e^{m_i - m_i^{\mathrm{new}}} \mathbf{O}_i + e^{\tilde{m}_{ij} - m_i^{\mathrm{new}}} \tilde{\mathbf{P}}_{ij} \mathbf{V}_j)$ to HBM.

    Write $\ell_i \leftarrow \ell_i^{\mathrm{new}}$, $m_i \leftarrow m_i^{\mathrm{new}}$ to HBM.

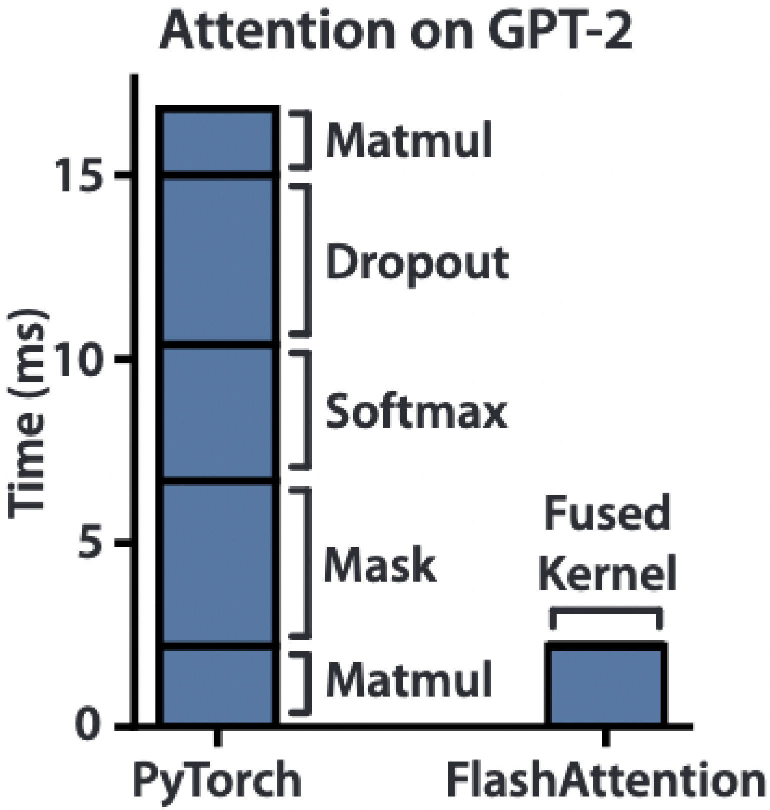
  end for

end for

Return $\mathbf{O}$.

$$\mathbf{S} = \mathbf{Q}\mathbf{K}^\top \in \mathbb{R}^{N\times N}, \quad \mathbf{P} = \mathrm{softmax}(\mathbf{S} \odot \mathbb{1}_{\tilde{\mathbf{M}}}) \in \mathbb{R}^{N\times N}, \quad \mathbf{O} = \mathbf{P}\mathbf{V} \in \mathbb{R}^{N\times d},$$

ПРОИЗВОДИТЕЛЬНОСТЬ

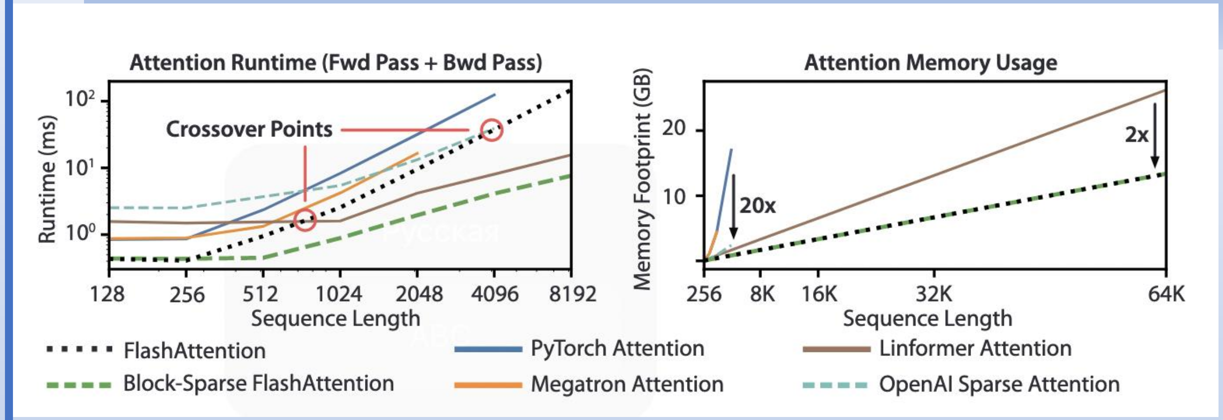


BERT Implementation	Training time (minutes)
Nvidia MLPerf 1.1 [56]	20.0 ± 1.5
FLASHATTENTION (ours)	**17.4** ± 1.4

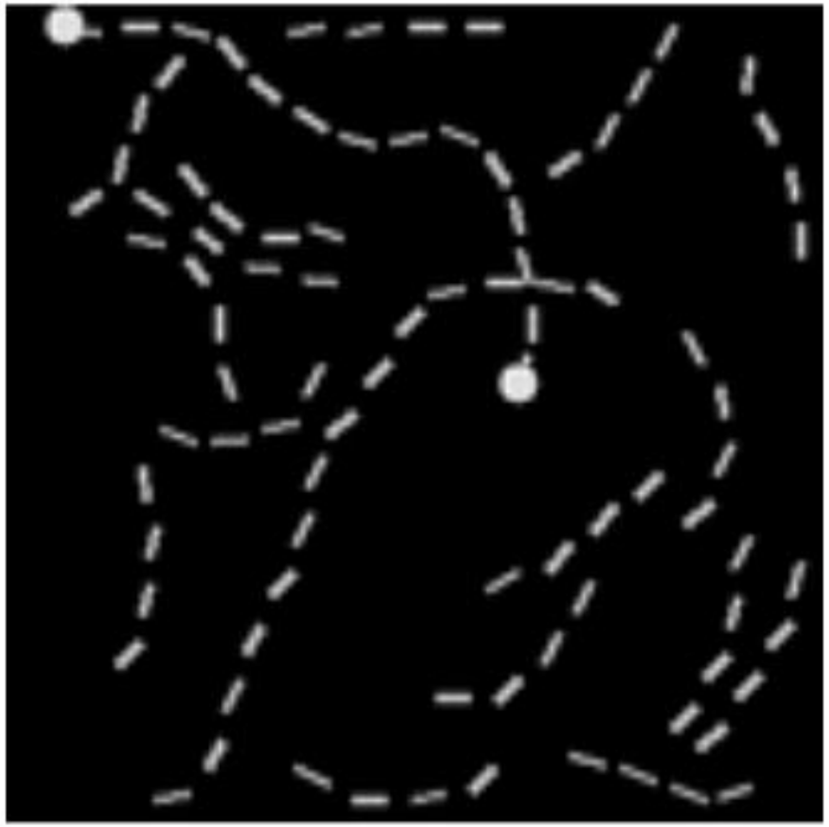
Model implementations	OpenWebText (ppl)	Training time (speedup)
GPT-2 small - Huggingface [84]	18.2	9.5 days (1.0×)
GPT-2 small - Megatron-LM [74]	18.2	4.7 days (2.0×)
GPT-2 small - FLASHATTENTION	18.2	**2.7 days (3.5×)**
GPT-2 medium - Huggingface [84]	14.2	21.0 days (1.0×)
GPT-2 medium - Megatron-LM [74]	14.3	11.5 days (1.8×)
GPT-2 medium - FLASHATTENTION	14.3	**6.9 days (3.0×)**

Models	ListOps	Text	Retrieval	Image	Pathfinder	Avg	Speedup
Transformer	36.0	63.6	81.6	42.3	72.7	59.3	-
FLASHATTENTION	37.6	63.9	81.4	43.5	72.7	59.8	2.4×
Block-sparse FLASHATTENTION	37.0	63.0	81.3	43.6	73.3	59.6	**2.8×**
Linformer [81]	35.6	55.9	77.7	37.8	67.6	54.9	2.5×
Linear Attention [48]	38.8	63.2	80.7	42.6	72.5	59.6	2.3×
Performer [11]	36.8	63.6	82.2	42.1	69.9	58.9	1.8×
Local Attention [77]	36.1	60.2	76.7	40.6	66.6	56.0	1.7×
Reformer [49]	36.5	63.8	78.5	39.6	69.4	57.6	1.3×
Smyrf [18]	36.1	64.1	79.0	39.6	70.5	57.9	1.7×

ЕЩЁ МЕТРИКИ



Model	Path-X	Path-256
Transformer	✗	✗
Linformer [81]	✗	✗
Linear Attention [48]	✗	✗
Performer [11]	✗	✗
Local Attention [77]	✗	✗
Reformer [49]	✗	✗
SMYRF [18]	✗	✗
FlashAttention	**61.4**	✗
Block-sparse FlashAttention	56.0	**63.1**



(a) A positive example.

FLASHATTENTION 2

УЛУЧШЕНИЯ:

1) убраны промежуточные нормализации softmax;

2) для backward pass хранится на один массив меньше;

3) другой порядок циклов и дополнительное распараллеливание по длине последовательности (в первом алгоритме уже используется распараллеливание по количеству голов и размеру батча);

4) синхронизация между warps не нужна.

Algorithm 1 FlashAttention-2 forward pass

Require: Matrices $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{N\times d}$ in HBM, block sizes B_c, B_r.

Divide $\mathbf{Q}$ into $T_r = \left\lceil \frac{N}{B_r} \right\rceil$ blocks $\mathbf{Q}_1, \dots, \mathbf{Q}_{T_r}$ of size $B_r \times d$ each, and divide $\mathbf{K}, \mathbf{V}$ in to $T_c = \left\lceil \frac{N}{B_c} \right\rceil$ blocks $\mathbf{K}_1, \dots, \mathbf{K}_{T_c}$ and $\mathbf{V}_1, \dots, \mathbf{V}_{T_c}$, of size $B_c \times d$ each.

Divide the output $\mathbf{O} \in \mathbb{R}^{N\times d}$ into T_r blocks $\mathbf{O}_i, \dots, \mathbf{O}_{T_r}$ of size $B_r \times d$ each, and divide the logsumexp L into T_r blocks $L_i, \dots, L_{T_r}$ of size B_r each.

for $1 \le i \le T_r$ **do**

  Load $\mathbf{Q}_i$ from HBM to on-chip SRAM.

  On chip, initialize $\mathbf{O}_i^{(0)} = (0)_{B_r\times d} \in \mathbb{R}^{B_r\times d}, \ell_i^{(0)} = (0)_{B_r} \in \mathbb{R}^{B_r}, m_i^{(0)} = (-\infty)_{B_r} \in \mathbb{R}^{B_r}$.

  for $1 \le j \le T_c$ **do**

    Load $\mathbf{K}_j, \mathbf{V}_j$ from HBM to on-chip SRAM.

    On chip, compute $\mathbf{S}_i^{(j)} = \mathbf{Q}_i \mathbf{K}_j^T \in \mathbb{R}^{B_r\times B_c}$.

    On chip, compute $m_i^{(j)} = \max(m_i^{(j-1)}, \mathrm{rowmax}(\mathbf{S}_i^{(j)})) \in \mathbb{R}^{B_r}$, $\tilde{\mathbf{P}}_i^{(j)} = \exp(\mathbf{S}_i^{(j)} - m_i^{(j)}) \in \mathbb{R}^{B_r\times B_c}$ (pointwise), $\ell_i^{(j)} = e^{m_i^{j-1} - m_i^{(j)}} \ell_i^{(j-1)} + \mathrm{rowsum}(\tilde{\mathbf{P}}_i^{(j)}) \in \mathbb{R}^{B_r}$.

    On chip, compute $\mathbf{O}_i^{(j)} = \mathrm{diag}(e^{m_i^{(j-1)} - m_i^{(j)}})^{-1} \mathbf{O}_i^{(j-1)} + \tilde{\mathbf{P}}_i^{(j)} \mathbf{V}_j$.

  end for

  On chip, compute $\mathbf{O}_i = \mathrm{diag}(\ell_i^{(T_c)})^{-1} \mathbf{O}_i^{(T_c)}$.

  On chip, compute $L_i = m_i^{(T_c)} + \log(\ell_i^{(T_c)})$.

  Write $\mathbf{O}_i$ to HBM as the i-th block of $\mathbf{O}$.

  Write L_i to HBM as the i-th block of L.

end for

Return the output $\mathbf{O}$ and the logsumexp L.

FLASHATTENTION 2

Не требуется синхронизация между warps.

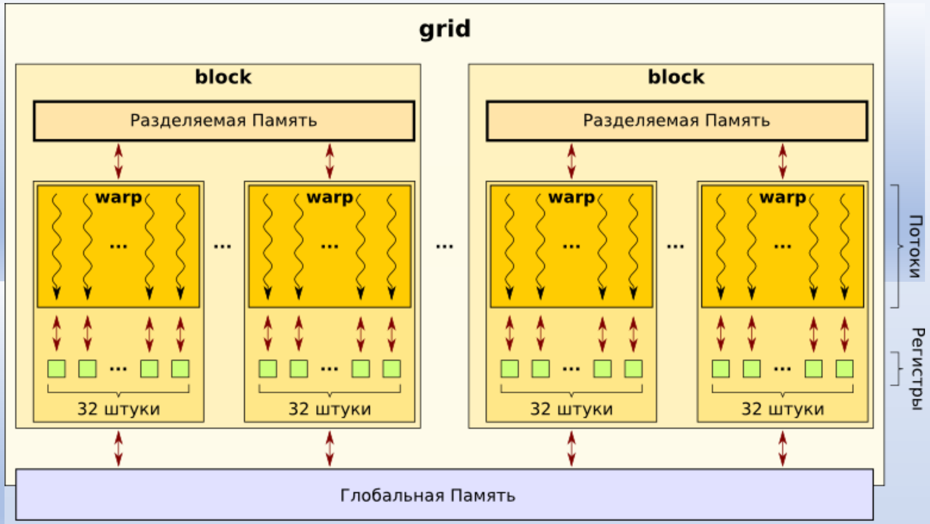
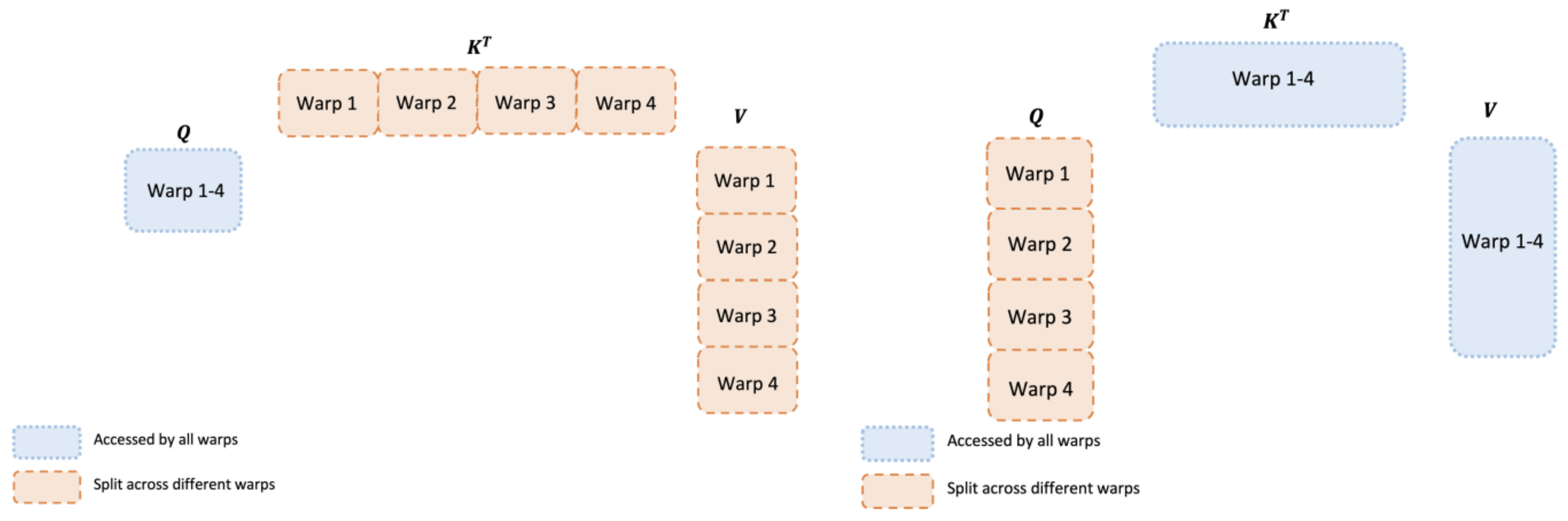


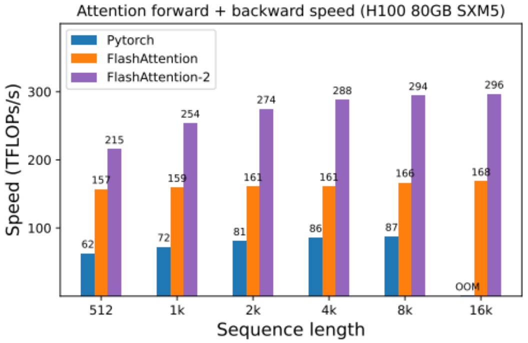
Рис. 1. Иерархия потоков и доступ к частям памяти.



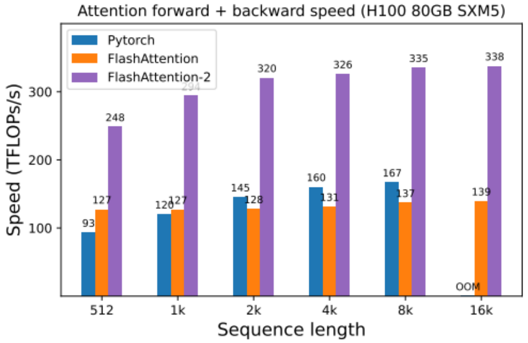
(a) FlashAttention (b) FlashAttention-2

ПРОИЗВОДИТЕЛЬНОСТЬ

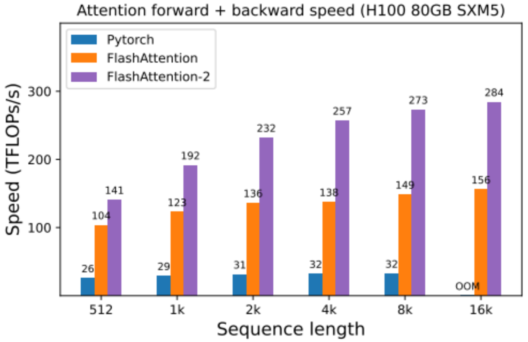
Авторы оставляют на будущее возможность получения дополнительного прироста скорости, сопоставимого с полученным с помощью FlashAttention 2, с использованием новых архитектурных особенностей видеокарт.



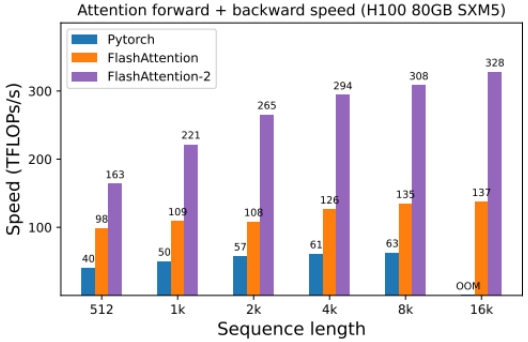
(a) Without causal mask, head dimension 64



(b) Without causal mask, head dimension 128

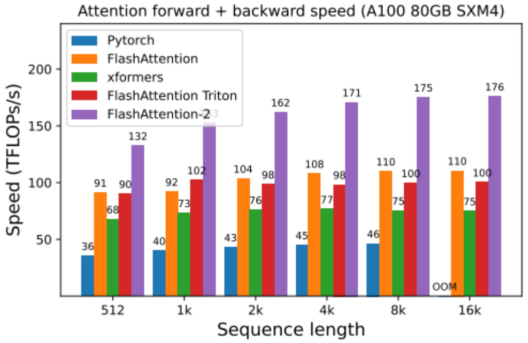


(c) With causal mask, head dimension 64

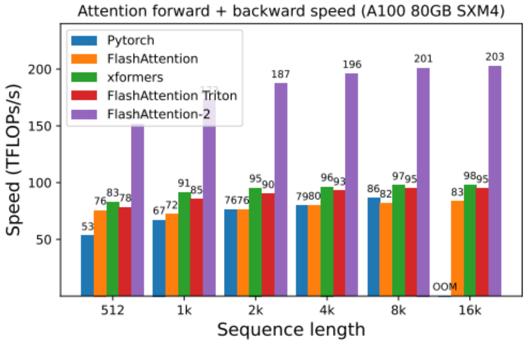


(d) With causal mask, head dimension 128

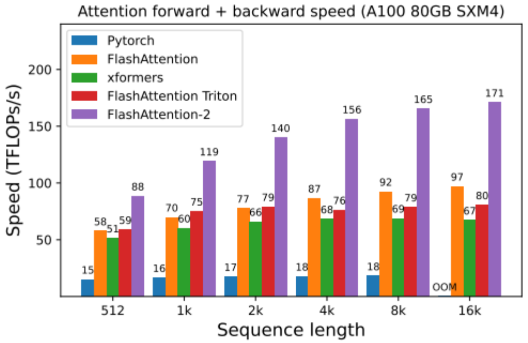
Figure 7: Attention forward + backward speed on H100 GPU



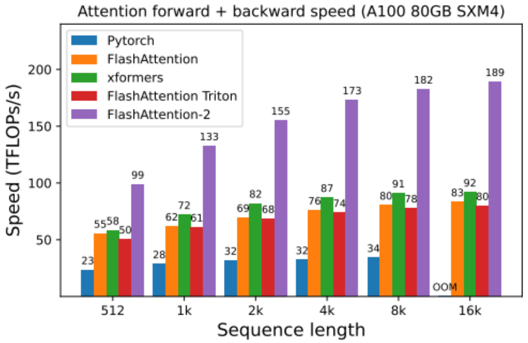
(a) Without causal mask, head dimension 64



(b) Without causal mask, head dimension 128



(c) With causal mask, head dimension 64



(d) With causal mask, head dimension 128

Figure 4: Attention forward + backward speed on A100 GPU

Model	Without FlashAttention	FlashAttention	FlashAttention-2
GPT3-1.3B 2k context	142 TFLOPs/s	189 TFLOPs/s	196 TFLOPs/s
GPT3-1.3B 8k context	72 TFLOPS/s	170 TFLOPs/s	220 TFLOPs/s
GPT3-2.7B 2k context	149 TFLOPs/s	189 TFLOPs/s	205 TFLOPs/s
GPT3-2.7B 8k context	80 TFLOPs/s	175 TFLOPs/s	225 TFLOPs/s

ССЫЛКИ

1. FlashAttention: https://arxiv.org/abs/2205.14135.

2. FlashAttention 2: https://arxiv.org/abs/2307.08691.

3. Небольшая статья про FlashAttention: https://habr.com/ru/articles/669506.