Реализация triton кернелей для квантизации весов в LLM и инференса квантизованной модели

Куляскин Михаил и Цыбикжапов Даши

Цель работы

Показать, что для большой языковой модели Llama-3.2-1B-Instruct можно:

1. Сильно уменьшить память под веса, квантовав их до 4 бит (int4),

2. Ускорить матричные умножения за счёт более компактного формата и специализированных Triton-кернелов,

3. При этом сохранить разумный уровень качества, измеряемый перплексией на датасете WikiText-2.

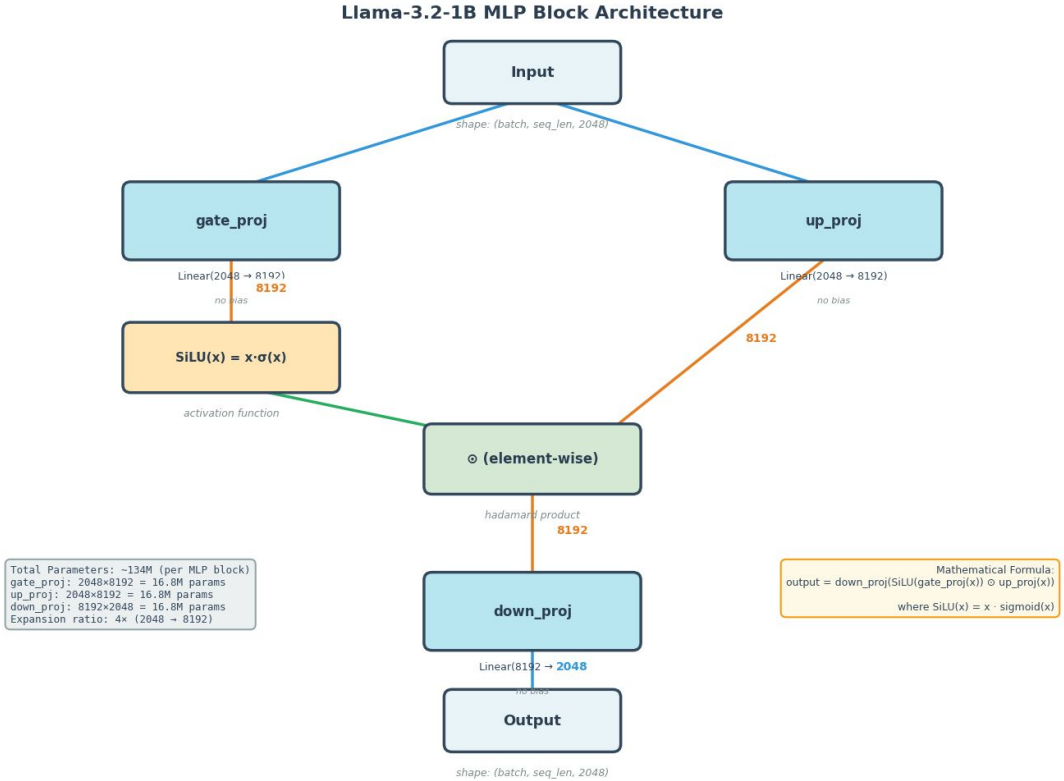
Поэтому надо реализовать:

1. свой кернел квантизации fp16 → int4 с упаковкой,

2. свой matmul-кернел BF16×INT4 (X16 @ W4ᵀ),

3. квантизованный линейный слой, который встроим в Llama-3.2-1B-Instruct, и далее измерить скорость и перплексию до и после квантования.

Архитектура LLaMA и где мы ускоряем модель



Основное время и память — в линейных слоях

Что такое квантизация и почему int4

БЫЛО:

16 бит	16 бит	16 бит	16 бит	16 бит	16 бит

СТАЛО:

4 бит, 4 бит	4 бит, 4 бит	4 бит, 4 бит

16 бит переводим в 2×4 бита в 1 байте, плюс один scale на строку, получаем ~ в 4 раза меньше памяти на веса

ИТОГО ПОЛУЧАЕМ:
УМЕНЬШЕНИЕ ПАМЯТИ
БЫСТРОЕ ЧТЕНИЕ ИЗ ПАМЯТИ
НО!:
НЕБОЛЬШАЯ ПОТЕРЯ ТОЧНОСТИ

Row-wise симметричная int4-квантизация

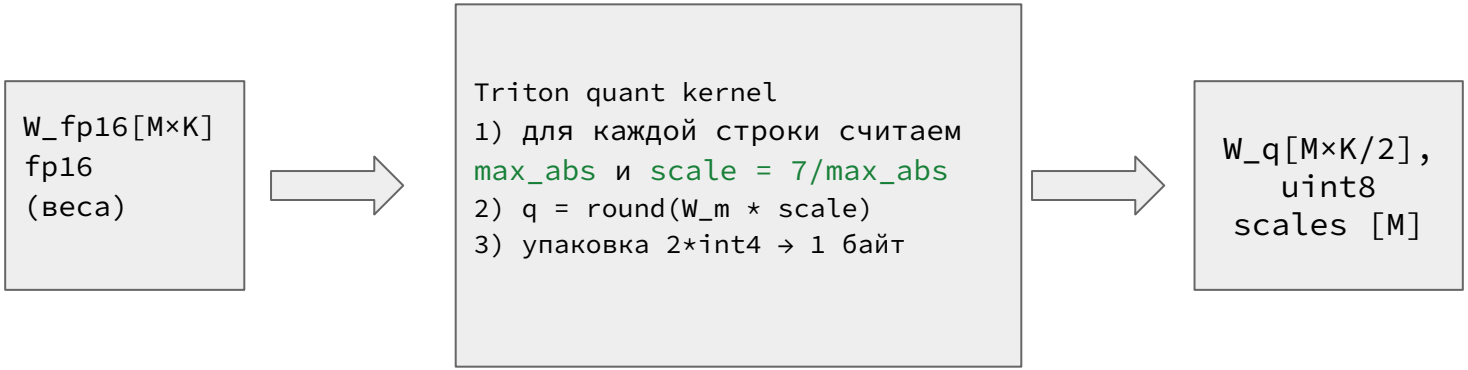
Per-tensor: один scale на всю матрицу, но сильно страдает качество, особенно при 4 битах.
Row-wise (per-channel): **один scale на строку / выходной нейрон**, получается чуть больше памяти (M скейлов), зато гораздо лучше качество - то, что мы и используем.
Group-wise / block-wise: отдельный scale на маленькие группы элементов (например, по 64–128 значений), ещё лучше качество, но сложнее реализация.

1) Квантуем матрицу весов линейного слоя.
2) Для **каждой строки WmWm** считаем свой масштаб
3) Симметрично квантуем в int4

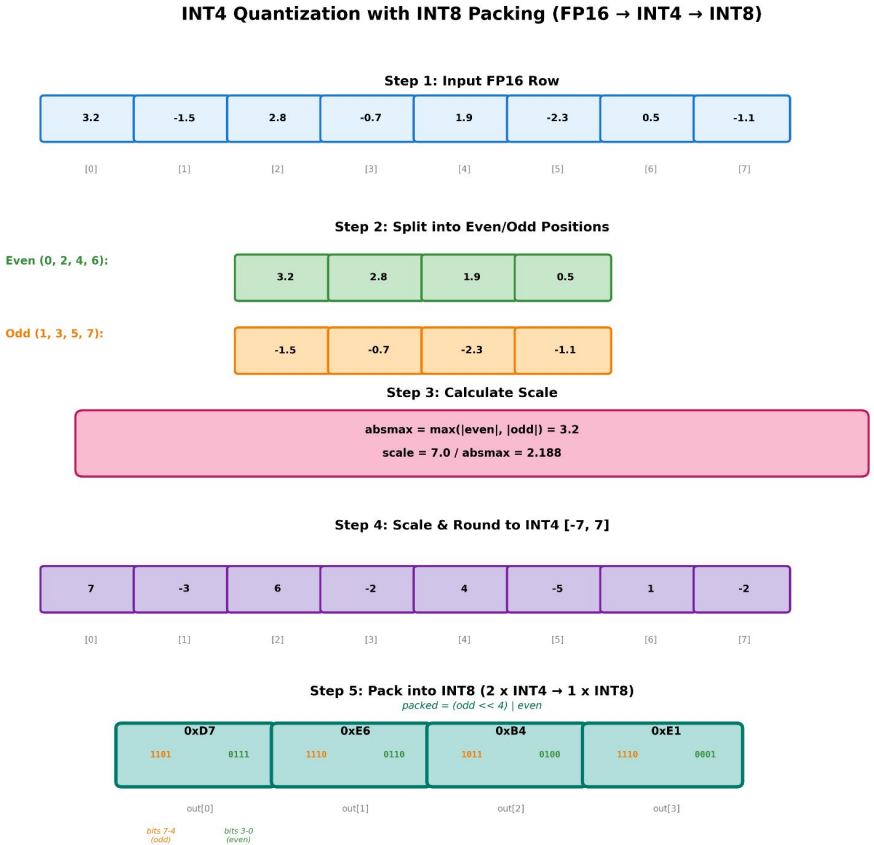
A Comprehensive Evaluation on Quantization Techniques for LLMs (2025)

https://arxiv.org/pdf/2507.17417
ZeroQuant (NeurIPS 2022)
https://arxiv.org/pdf/2206.01861

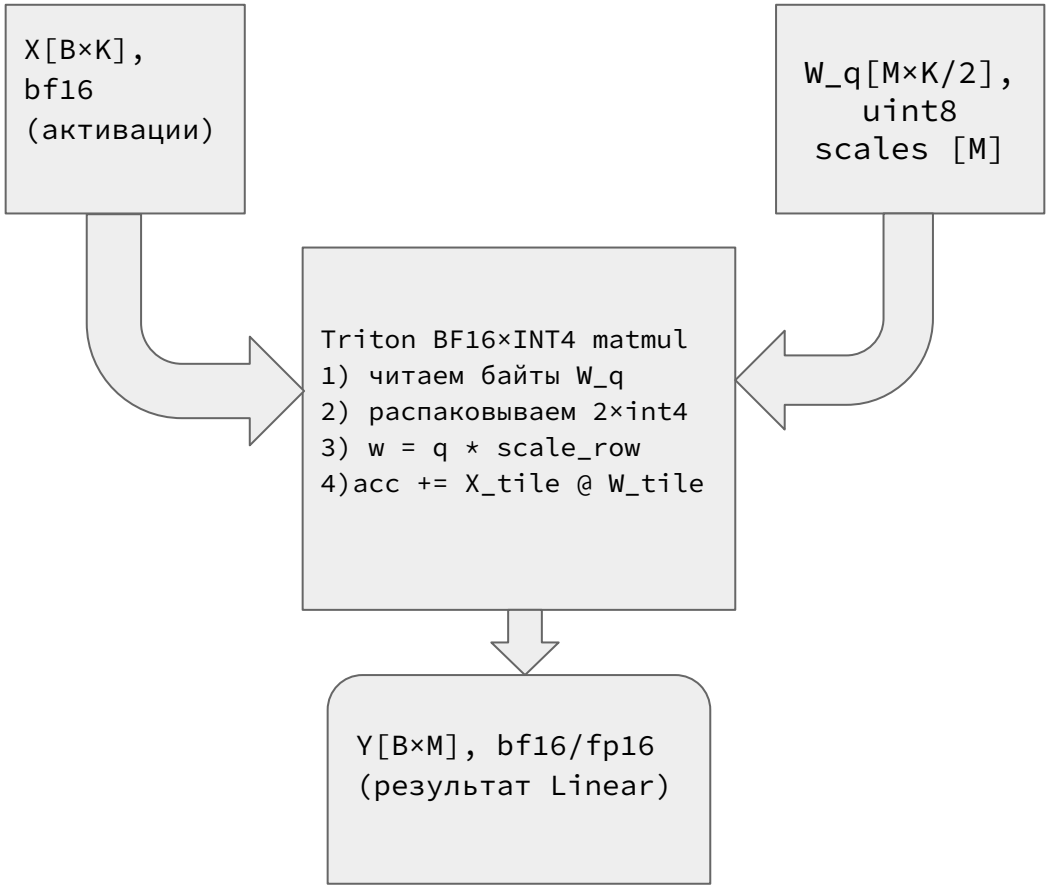
Triton-кернел квантизации



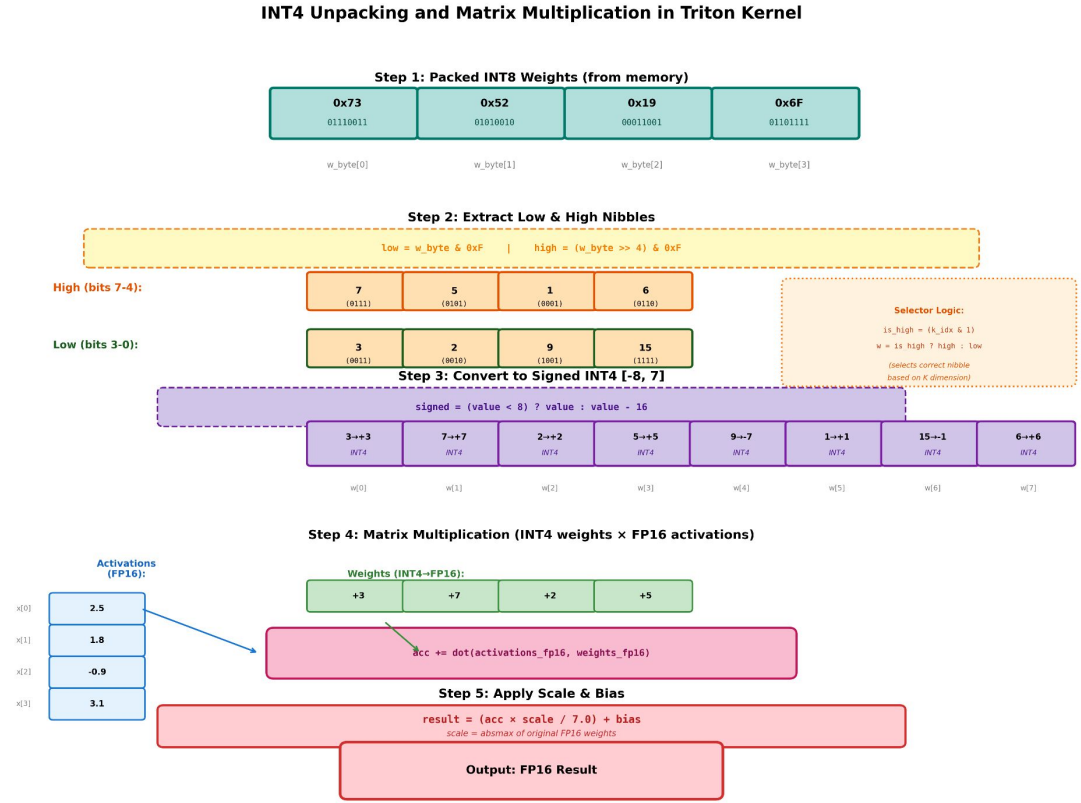
Упаковка int4 в память



Triton-кернел BF16×INT4 matmul



Распаковка int4



Встраивание в Llama-3.2-1B

Рекурсивно обходим модель, все **nn.Linear** заменяем на **QuantLinear** с квантизованными весами

```
def replace_linear_with_quantlinear(module: torch.nn.Module):
    for child_name, child in module.named_children():
        if isinstance(child, torch.nn.Linear):
            quant_linear = quantize_linear_module(child)
            setattr(module, child_name, quant_linear)
        else:
            replace_linear_with_quantlinear(child)

replace_linear_with_quantlinear(model)
```

[82]

```
model.model.layers[5].mlp.gate_proj.weight_scale
```

[89]

```
tensor([0.0957, 0.0957, 0.0645,  ..., 0.0767, 0.1133, 0.0591], device='cuda:0',
       dtype=torch.float16)
```

Квантизованный Linear: QuantLinear

`nn.Linear` –> `QuantLinear`

1) забираем веса, квантуем row-wise, храним `W_q` + `scale`
2) в `forward`: `[batch, seq, hidden]` → `[B, K]` → `matmul_int4` → `[B, M]` → `[batch, seq, M]`

```python
class QuantLinear(torch.nn.Module):
    def __init__(self, in_features: int, out_features: int,
                 weight_quant: torch.Tensor, weight_scale: torch.Tensor,
                 bias: torch.Tensor | None = None):
        super().__init__()
        self.in_features = in_features
        self.out_features = out_features
        self.weight_q = weight_quant
        self.weight_scale = weight_scale
        self.bias = bias

    def forward(self, x: torch.Tensor) -> torch.Tensor:
        orig_shape = x.shape
        x_2d = x.reshape(-1, orig_shape[-1])
        y_2d = matmul_int4_fused(
            x=x_2d,
            w_q=self.weight_q,
            w_scale=self.weight_scale,
            bias=self.bias,
        )
        y = y_2d.reshape(*orig_shape[:-1], self.out_features)
        return y
```

[81]

Эксперименты: скорость матмулов

Out[1]:

	type	memory_mb
0	bf16	32.0
1	int4	8.0

In [2]:
```
speed_results
```

Out[2]:

	B	int4_ms	fp16_ms	speedup
0	128	0.531	0.226	0.43
1	512	1.299	0.581	0.45
2	2048	4.806	2.884	0.60

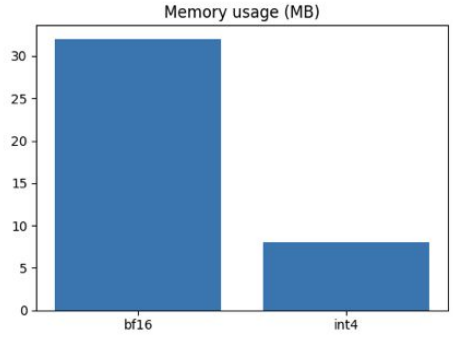
In [3]:
```
ppl_results
```

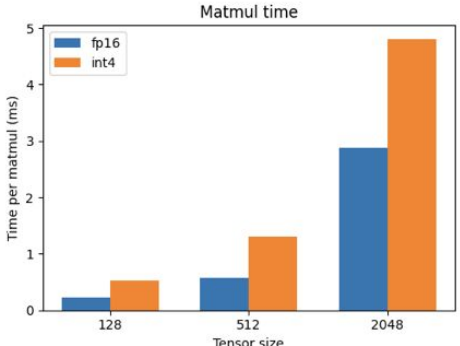
Out[3]:

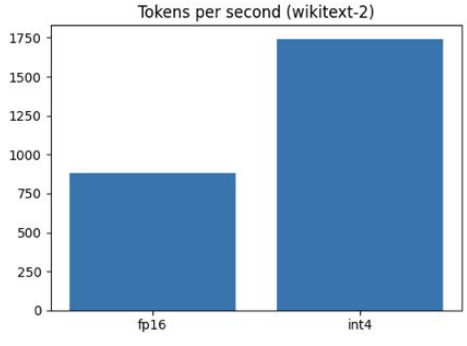
	model	perplexity	eval_time_s	tokens	tokens_per_sec
0	fp16	36.8581	284.20	249856	879.17
1	int4	71.0608	143.44	249856	1741.88

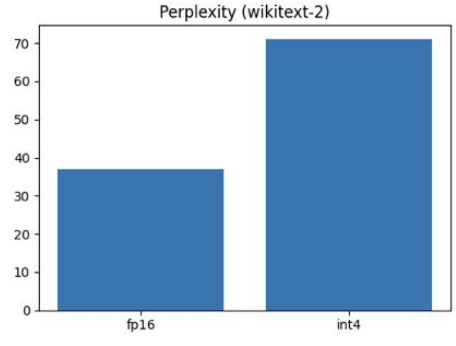
квантованная:
Perplexity: 71.0608
Eval time (s): 143.44 Tokens processed: 249856 Tokens/sec: 1741.88

неквантованная:
Perplexity: 36.8581
Eval time (s): 284.20 Tokens processed: 249856 Tokens/sec: 879.17









Итоги

Реализовали свои Triton-кернелы квантизации и matmul BF16×INT4
встроили их в Llama-3.2-1B через QuantLinear
показали, что можно:

- уменьшить память под веса ≈ в 4 раза,
- сохранить адекватную перплексию,
- ускорить инференс.