Generative Inference Algorithm Optimization

COMP6211J

Binhang Yuan

Announcement about the Presentation

RELAXED
SYSTEM LAB

- Evaluation rubric:
  - The talk organizes the material with good categorization (0-5 points);
  - The talk is well-motivated (0-5 points);
    -- “I found this presentation interesting .”
  - The talk is self-explained (0-5 points);
    -- “I learned something interesting from this presentation.”
  - The presenter answered the question appropriately (0-5 points).
- The presentation is in-person, you have to be physically here.
- Do not miss your session, otherwise you get 0 credit for the presentation.

Generative Inference

Recall Generative Inference Workflow

RELAXED SYSTEM LAB

- State-of-the-art implementation splits the computation to two phrases:
  - Prefill phase: the model takes a prompt sequence as input and engages in the generation of a key-value cache (KV cache) for each Transformer layer.
  - Decode phase: for each decode step, the model updates the KV cache and reuses the KV to compute the output.
- Analyze the arithmetic intensity:
  - Prefill phrase: arithmetic bounded;
  - Decode phrase: memory bounded.
- Assume the computation is in fp16, the concrete analysis in next two slides.
  - L is the input sequence length;
  - D is the model dimension;
  - Multi-head attention $D = n_H \times H$; H is the head dimension; n_h is the number of heads.

Prefill Phrase

RELAXED SYSTEM LAB

No.	Computation	Input	Output	Arithmetic Intensity
1	$Q = xW^Q$	$x \in \mathbb{R}^{L\times D}$, $W^Q \in \mathbb{R}^{D\times D}$	$Q \in \mathbb{R}^{L\times D}$	$\frac{L\times D^2}{L\times D+D^2+L\times D}=\frac{L\times D}{2L+D}$
2	$K = xW^K$	$x \in \mathbb{R}^{L\times D}$, $W^K \in \mathbb{R}^{D\times D}$	$K \in \mathbb{R}^{L\times D}$	$\frac{L\times D^2}{L\times D+D^2+L\times D}=\frac{L\times D}{2L+D}$
3	$V = xW^V$	$x \in \mathbb{R}^{L\times D}$, $W^V \in \mathbb{R}^{D\times D}$	$V \in \mathbb{R}^{L\times D}$	$\frac{L\times D^2}{L\times D+D^2+L\times D}=\frac{L\times D}{2L+D}$
4	$[Q_1, Q_2 \ldots, Q_{n_h}] = \text{Partition}_{-1}(Q)$	$Q \in \mathbb{R}^{L\times D}$	$Q_i \in \mathbb{R}^{L\times H}, i = 1, \ldots n_h$	-
5	$[K_1, K_2 \ldots, K_{n_h}] = \text{Partition}_{-1}(K)$	$K \in \mathbb{R}^{L\times D}$	$K_i \in \mathbb{R}^{L\times H}, i = 1, \ldots n_h$	-
6	$[V_1, V_2 \ldots, V_{n_h}] = \text{Partition}_{-1}(V)$	$V \in \mathbb{R}^{L\times D}$	$V_i \in \mathbb{R}^{L\times H}, i = 1, \ldots n_h$	-
7	$\text{Score}_i = \text{softmax}(\frac{Q_i K_i^T}{\sqrt{D}}), i = 1, \ldots n_h$	$Q_i, K_i \in \mathbb{R}^{L\times H}$	$\text{score}_i \in \mathbb{R}^{L\times L}$	$\frac{L^2\times H}{L\times H+L^2+L\times H}=\frac{L\times H}{2H+L}$
8	$Z_i = \text{score}_i\, V_i, i = 1, \ldots n_h$	$\text{score}_i \in \mathbb{R}^{L\times L}$, $V_i \in \mathbb{R}^{L\times H}$	$Z_i \in \mathbb{R}^{L\times H}$	$\frac{L^2\times H}{L\times H+L^2+L\times H}=\frac{L\times H}{2H+L}$
9	$Z = \text{Merge}_{-1}([Z_1, Z_2 \ldots, Z_{n_h}])$	$Z_i \in \mathbb{R}^{L\times H}, i = 1, \ldots n_h$	$Z \in \mathbb{R}^{L\times D}$	-
10	$\text{Out} = ZW^O$	$Z \in \mathbb{R}^{L\times D}$, $W^O \in \mathbb{R}^{D\times D}$	$\text{Out} \in \mathbb{R}^{L\times D}$	$\frac{L\times D^2}{L\times D+D^2+L\times D}=\frac{L\times D}{2L+D}$
11	$A = \text{Out}\, W^1$	$\text{Out} \in \mathbb{R}^{L\times D}$, $W^1 \in \mathbb{R}^{D\times 4D}$	$A \in \mathbb{R}^{L\times 4D}$	$\frac{4L\times D^2}{L\times D+4D^2+L\times 4D}=\frac{4L\times D}{5L+4D}$
12	$A' = \text{relu}(A)$	$A \in \mathbb{R}^{L\times 4D}$	$A' \in \mathbb{R}^{L\times 4D}$	-
13	$x' = A'W^2$	$A' \in \mathbb{R}^{L\times 4D}$, $W^2 \in \mathbb{R}^{4D\times D}$	$x' \in \mathbb{R}^{L\times D}$	$\frac{4L\times D^2}{L\times D+4D^2+L\times 4D}=\frac{4L\times D}{5L+4D}$

Decoding Phrase

RELAXED SYSTEM LAB

No	Computationt	Input	Output	Arithmetic Intensity
1	$Q = Q_d = tW^Q$	$t \in \mathbb{R}^{1\times D}$, $W^Q \in \mathbb{R}^{D\times D}$	$Q, Q_d \in \mathbb{R}^{1\times D}$	$\frac{1\times D^2}{1\times D+D^2+1\times D}=\frac{D}{2+D}$
2	$K_d = tW^K$	$t \in \mathbb{R}^{1\times D}$, $W^K \in \mathbb{R}^{D\times D}$	$K_d \in \mathbb{R}^{1\times D}$	$\frac{1\times D^2}{1\times D+D^2+1\times D}=\frac{D}{2+D}$
3	$K = \text{concat}(K_{\text{cache}}, K_d)$	$K_{\text{cache}} \in \mathbb{R}^{L\times D}, K_d \in \mathbb{R}^{1\times D}$	$K \in \mathbb{R}^{(L+1)\times D}$	-
4	$V_d = tW^V$	$t \in \mathbb{R}^{1\times D}$, $W^V \in \mathbb{R}^{D\times D}$	$V_d \in \mathbb{R}^{1\times D}$	$\frac{1\times D^2}{1\times D+D^2+1\times D}=\frac{D}{2+D}$
5	$V = \text{concat}(V_{\text{cache}}, V_d)$	$V_{\text{cache}} \in \mathbb{R}^{L\times D}, V_d \in \mathbb{R}^{1\times D}$	$V \in \mathbb{R}^{(L+1)\times D}$	-
6	$[Q_1, Q_2 \dots, Q_{n_h}] = \text{Partition}_{-1}(Q)$	$Q \in \mathbb{R}^{1\times D}$	$Q_i \in \mathbb{R}^{1\times H}, i = 1, \dots n_h$	-
7	$[K_1, K_2 \dots, K_{n_h}] = \text{Partition}_{-1}(K)$	$K \in \mathbb{R}^{(L+1)\times D}$	$K_i \in \mathbb{R}^{(L+1)\times H}, i = 1, \dots n_h$	-
8	$[V_1, V_2 \dots, V_{n_h}] = \text{Partition}_{-1}(V)$	$V \in \mathbb{R}^{(L+1)\times D}$	$V_i \in \mathbb{R}^{(L+1)\times H}, i = 1, \dots n_h$	-
9	$\text{Score}_i = \text{softmax}(\frac{Q_i K_i^T}{\sqrt{D}}), i = 1, \dots n_h$	$Q_i \in \mathbb{R}^{1\times H}, K_i \in \mathbb{R}^{(L+1)\times H}$	$\text{score}_i \in \mathbb{R}^{1\times (L+1)}$	$\frac{1\times (L+1)\times H}{1\times H+(L+1)\times H+L+1}=\frac{H+LH}{2H+L+LH+1}$
10	$Z_i = \text{score}_i\, V_i, i = 1, \dots n_h$	$\text{score}_i \in \mathbb{R}^{1\times (L+1)}, V_i \in \mathbb{R}^{(L+1)\times H}$	$Z_i \in \mathbb{R}^{1\times H}$	$\frac{(L+1)\times H}{L+1+(L+1)\times H+1\times H}=\frac{H+LH}{2H+L+LH+1}$
11	$Z = \text{Merge}_{-1}([Z_1, Z_2 \dots, Z_{n_h}])$	$Z_i \in \mathbb{R}^{1\times H}, i = 1, \dots n_h$	$Z \in \mathbb{R}^{1\times D}$	-
12	$\text{Out} = ZW^O$	$Z \in \mathbb{R}^{1\times D}$, $W^O \in \mathbb{R}^{D\times D}$	$\text{Out} \in \mathbb{R}^{1\times D}$	$\frac{1\times D^2}{1\times D+D^2+1\times D}=\frac{D}{2+D}$
13	$A = \text{Out}\, W^1$	$\text{Out} \in \mathbb{R}^{1\times D}$, $W^1 \in \mathbb{R}^{D\times 4D}$	$A \in \mathbb{R}^{1\times 4D}$	$\frac{1\times 4D^2}{1\times D+4D^2+1\times 4D}=\frac{4D}{5+4D}$
14	$A' = \text{relu}(A)$	$A \in \mathbb{R}^{1\times 4D}$	$A' \in \mathbb{R}^{1\times 4D}$	-
15	$t' = A'W^2$	$A' \in \mathbb{R}^{1\times 4D}$, $W^2 \in \mathbb{R}^{4D\times D}$	$t' \in \mathbb{R}^{1\times D}$	$\frac{1\times 4D^2}{1\times 4D+4D^2+1\times D}=\frac{4D}{5+4D}$

Optimization Goal

RELAXED
SYSTEM LAB

- Decrease the I/O volume:
  - Model compression, i.e., quantization;
  - Knowledge distillation.
  - KV cache optimization.
- Increase the computation load for each generation step (next time) :
  - Speculative decoding;
  - Disaggregate inference framework.

Model Quantization

Quantization

- Quantization transforms the floating-point values in original LLMs into integers or other discrete forms to reduce memory requirements and computational complexity.
  - int4/int8 takes 4/8 bits;
  - GPU also has a higher throughput for int4/int8 computation compared with fp16.
- How do we get the quantized model?
  - Post-Training Quantization (PTQ).
  - Quantization-Aware Training (QAT).

Preliminary of Quantization

RELAXED SYSTEM LAB

- How to quantize a float point number?
- **Absolute maximum (absmax) quantization**:
  - The original number is divided by the absolute maximum value of the tensor and multiplied by a scaling factor (127) to map inputs into the range [-127, 127].

$$\boldsymbol{X}_{\text{quant}} = \text{round}\left(\frac{127}{\max|\boldsymbol{X}|} \cdot \boldsymbol{X}\right)$$

  - To retrieve the original FP16 values, the INT8 number is divided by the quantization factor.

$$\boldsymbol{X}_{\text{dequant}} = \frac{\max|X|}{127} \cdot \boldsymbol{X}_{\text{quant}}$$

  - For example, suppose we have an absolution maximum value of 3.2. A weight of 0.1 would be quantized to round$(0.1 \times 127/3.2) = 4$. If we want to dequantize it, we would get $4 \times 3.2/127 = 0.1008$, which implies an error of 0.008.

Preliminary of Quantization

RELAXED SYSTEM LAB

- **Zero-point quantization**:
  - The input values are first scaled by the total range of values (255) divided by the difference between the maximum and minimum values. This distribution is then shifted by the zero-point to map it into the range [-128, 127]. First, we calculate the scale factor and the zero-point value:

$$\text{scale} = \frac{255}{\max(\boldsymbol{X}) - \min(\boldsymbol{X})}$$

$$\text{zeropoint} = -\text{round}(\text{scale} \cdot \min(\boldsymbol{X})) - 128$$

  - We can use these variables to quantize:

$$\boldsymbol{X}_{\text{quant}} = \text{round}(\text{scale} \cdot \boldsymbol{X} + \text{zeropoint})$$

  - We can use these variables to dequantize.

$$\boldsymbol{X}_{\text{dequant}} = \frac{\boldsymbol{X}_{\text{quant}} - \text{zeropoint}}{\text{scale}}$$

  - For example, suppose we have a maximum value of 3.2 and a minimum value of -3.0. We can calculate the scale is 255/(3.2 + 3.0) = 41.13 and the zero-point -round(41.13 × -3.0) - 128 = 123 -128 = -5, so our previous weight of 0.1 would be quantized to round(41.13 × 0.1 -5) = -1.

Post-Training Quantization (PTQ)

RELAXED SYSTEM LAB

- Post-training quantization (PTQ) quantizes a pre-trained model using moderate resources, such as a calibration dataset and a few hours of computation.
- PTQ reduces the computational precision of model weights and even activations into either INT8 or INT4 by using custom CUDA kernels for efficiency benefits.
- PTQ is an effective method for enhancing the efficiency of LLMs. It offers a straightforward solution that avoids major modifications or extensive additional training.
- Quantization targets:
  - Weight-only quantization;
  - Weights and activations quantization.
- It is necessary to acknowledge that PTQ can lead to a certain degree of precision loss due to the quantization process.

PTQ – Weight-only Quantization

RELAXED SYSTEM LAB

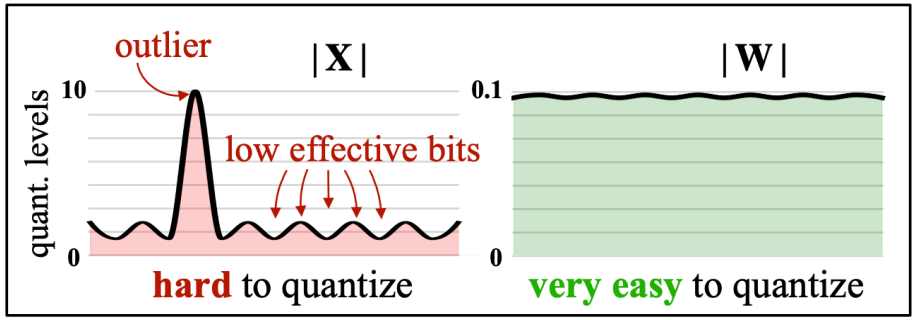
- There are many different methods for weight-only quantization.
- An example approach is **GPTQ** [https://arxiv.org/pdf/2210.17323v1.pdf]
  - GPTQ adopts a mixed int4/fp16 quantization scheme where weights are quantized as int4 while activations remain in float16.
  - During inference, weights are dequantized on the fly and the actual compute is performed in float16.
  - How does GPTQ tackle the quantization problem? (Not required, Details can be found in the paper)
    - **Layerwise Quantization:** aims to find quantized values that minimize the error at the output.

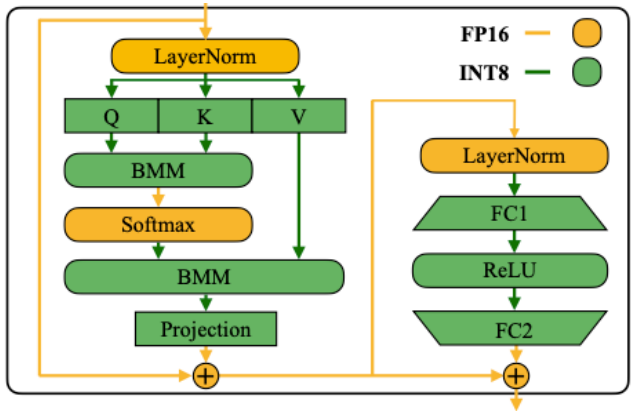
$$\text{argmin}_{\widehat{W}} \|WX - \widehat{W}X\|$$

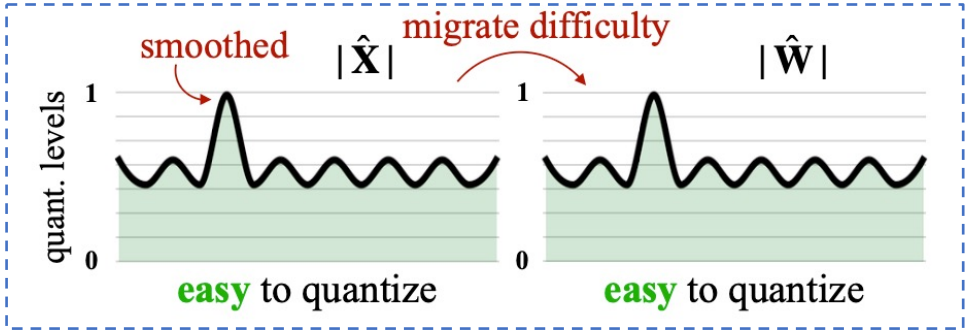
PTQ -Weights & Activations Quantization

RELAXED SYSTEM LAB

- *Why is quantizing activations difficult?*
  - Outliers make activation quantization difficult.
- An example approach is SmoothQuant [https://proceedings.mlr.press/v202/xiao23c/xiao23c.pdf].
  - SmoothQuant enables 8-bit weight, 8-bit activation (W8A8) quantization.
  - All the compute-intensive operators, such as the linear layers, use int8 operations. But still not all computations are in int8.
  - How does SmoothQuant solve the problem?
    - Observation: fixed channels have outliers, and the outlier channels are persistently large.
    - Solution: smooth the outlier channels in activations by migrating their magnitudes into the following weights.



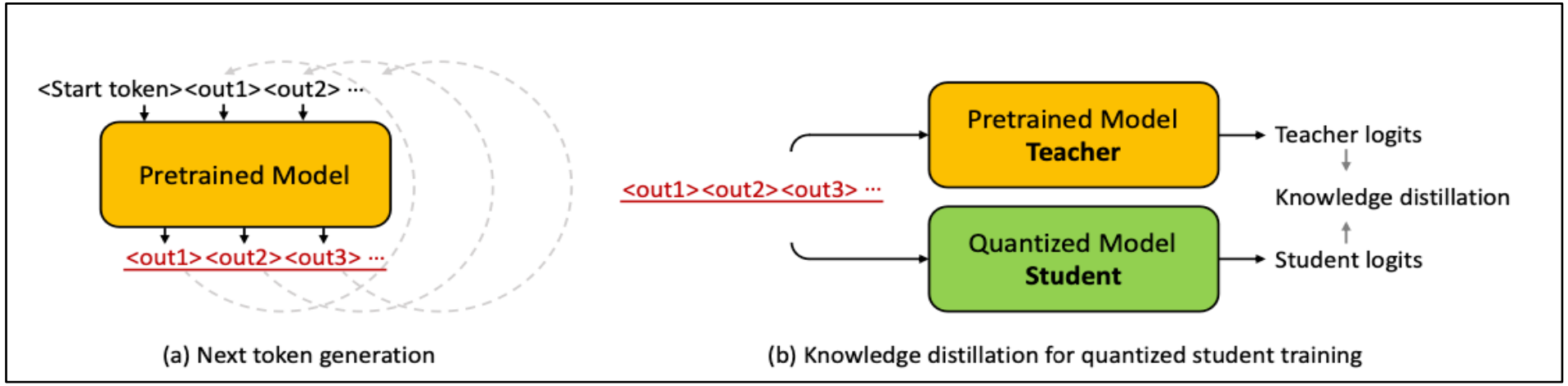




Quantization-Aware Training (QAT)

RELAXED SYSTEM LAB

- Quantization-Aware Training (QAT): the quantization process is seamlessly integrated into the training of LLMs.
- QAT can adapt to low-precision representations and thus mitigating precision loss.
- QAT require more computation than PTQ.
- An example approach is LLM-QAT [https://arxiv.org/abs/2305.17888]
  - A data-free distillation method: LLM-QAT generates data from the pretrained model with next token generation, which is sampled from top-k candidates. Then it uses the generated data as input and the teacher model prediction as label to guide quantized model finetuning.



Knowledge Distillation

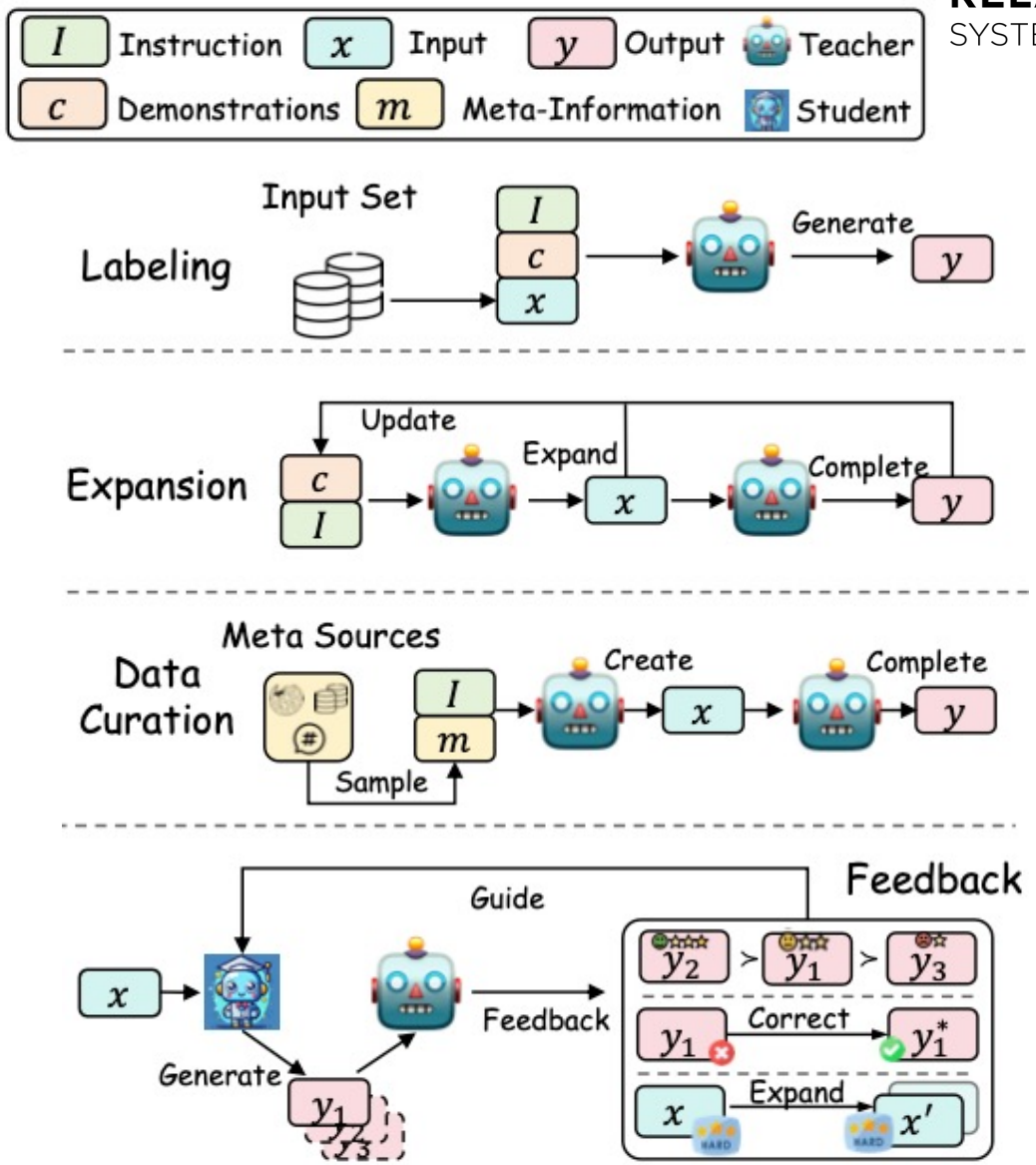
Knowledge Distillation

RELAXED SYSTEM LAB

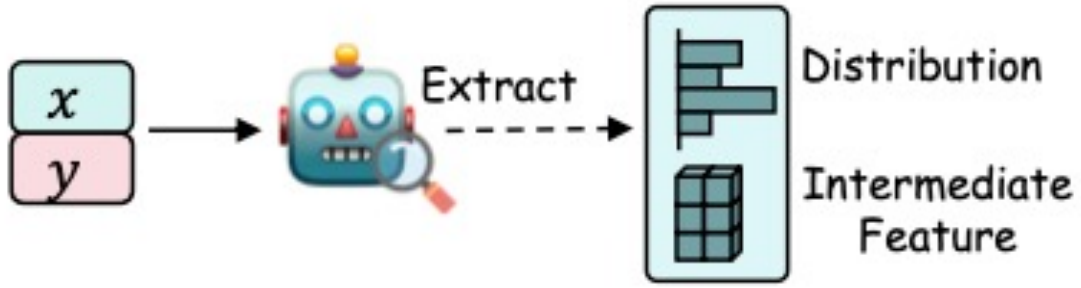
- Knowledge distillation is a technique that facilitates the transfer of capabilities from a larger model (referred to as the "*teacher model*") to a smaller model (referred to as the "*student model*").
- Knowledge distillation allows the smaller model to perform tasks with proficiency similar to the larger model but with reduced computational resources.
- Two main categories:
  - **Black-box distillation**: focuses on replicating the output behavior of the teacher model. The student model learns solely from the input-output pairings produced by the teacher without any insight into its internal operations.
  - **White-box distillation:** the internal workings and architecture of the teacher model are fully accessible and utilized during the distillation process.

Black-box Distillation

- In white-box distillation, there are different knowledge elicitation methods for teacher LLMs:
  - *Labeling*: The teacher generates the output from the input;
  - *Expansion*: The teacher generates samples similar to the given demonstrations through in-context learning;
  - *Data Curation*: The teacher synthesizes data according to meta-information, such as a topic or an entity;
  - *Feedback*: The teacher provides feedback on the student's generations, such as preferences, corrections, and expansions of challenging samples.



White-box Distillation



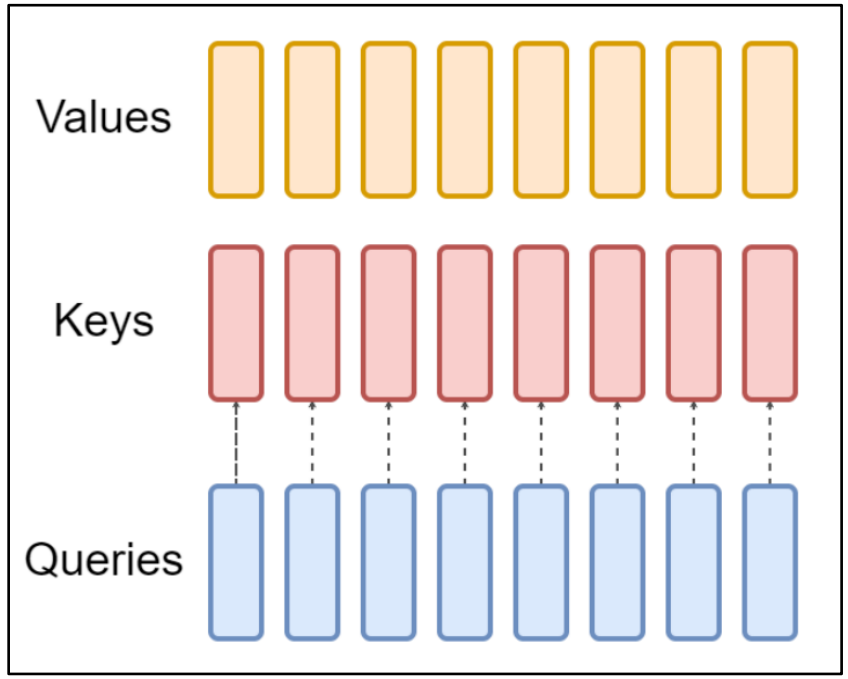
- White-box distillation involves leveraging the output distributions or intermediate features from teacher LLMs.
  - *Output distributions*: the teacher model labels a fixed dataset of sequences with token-level probability distributions. The KL divergence can be used as the loss function.
  - *Intermediate features*: To leverage the rich semantic and syntactic knowledge in intermediate layers of the teacher model, layer-wise distillation can align the student's hidden representations with those of the teacher at each layer.

KV Cache Optimization

Memory Footprint of Inference Computation

RELAXED SYSTEM LAB

- Suppose:
  - L is the max context-length;
  - D is the model dimension;
  - Multi-head attention:
    - $D = n_H \times H$
    - H is the head dimension;
    - n_h is the number of heads.
  - Everything computation is in FP16
- For each transformer layer:
  - Model parameters:
    - $2\times(D^2 + D^2 + D^2 + D^2 + 4D^2 + 4D^2) = 24\times D^2$
  - KV cache:
    - $2\times(LD + LD) = 4LD$
  - When $L > 6D$, KV cache can be the new bottleneck.



KV Cache Optimization

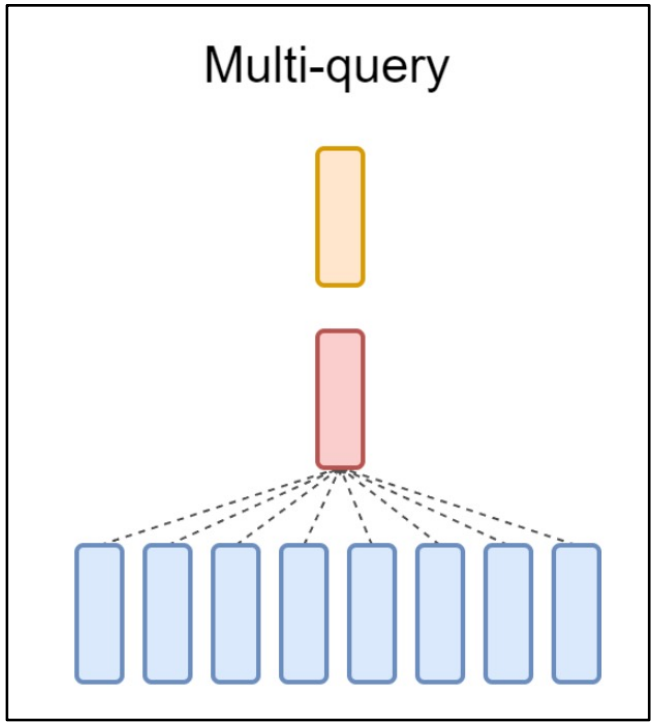
RELAXED
SYSTEM LAB

- Reduce the KV cache by modifying the multi-head attention architecture:
  - Group query attention (GQA)
- KV cache compression:
  - KV cache quantization;
  - Reserve more important tokens;
  - Sparsity at the fine granularity.

Multi-Query Attention (MQA)

RELAXED SYSTEM LAB

- The idea is simple yet effective:
  - Use multiple query heads but only a single key and value head.
- Convert MHA to MQA:
  - Initialize: the weight matrices for key and value heads are mean pooled into single weight matrices;
  - Train the model with the original recipe for a small portion of steps.



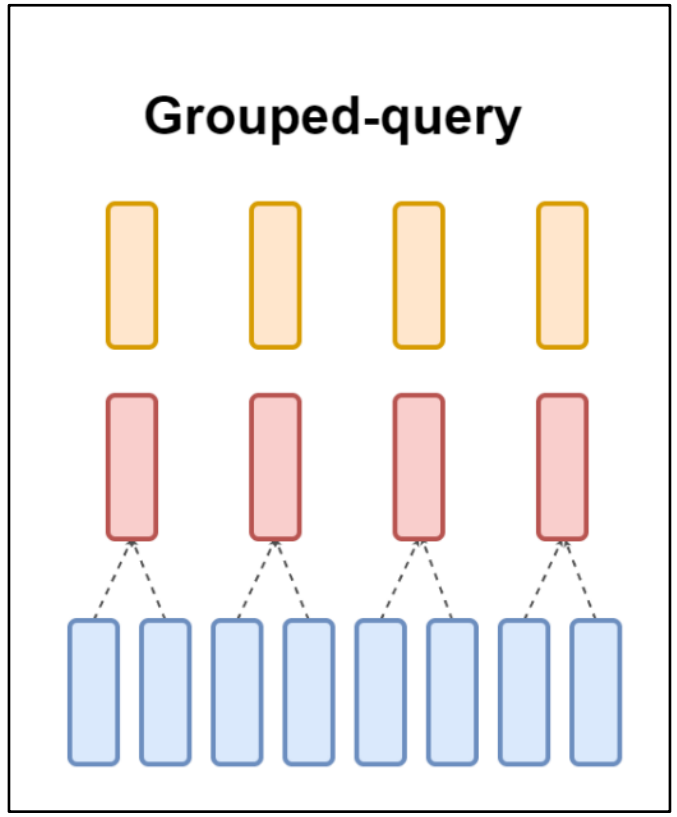
Computation	Input	Output
$Q = xW^Q$	$x \in \mathbb{R}^{L\times D}, W^Q \in \mathbb{R}^{D\times D}$	$Q \in \mathbb{R}^{L\times D}$
$K = xW^K$	$x \in \mathbb{R}^{L\times D}, W^K \in \mathbb{R}^{D\times H}$	$K \in \mathbb{R}^{L\times H}$
$V = xW^V$	$x \in \mathbb{R}^{L\times D}, W^V \in \mathbb{R}^{D\times H}$	$V \in \mathbb{R}^{L\times H}$
$[Q_1, Q_2 \dots, Q_{n_h}] = \text{Partition}_{-1}(Q)$	$Q \in \mathbb{R}^{L\times D}$	$Q_i \in \mathbb{R}^{L\times H}, i = 1, \dots n_h$
$\text{Score}_i = \text{softmax}(\frac{Q_i K^T}{\sqrt{D}}), i = 1, \dots n_h$	$Q_i, K \in \mathbb{R}^{L\times H}$	$\text{score}_i \in \mathbb{R}^{L\times L}$
$Z_i = \text{score}_i V, i = 1, \dots n_h$	$\text{score}_i \in \mathbb{R}^{L\times L}, V \in \mathbb{R}^{L\times H}$	$Z_i \in \mathbb{R}^{L\times H}$

Group-Query Attention (GQA)

RELAXED SYSTEM LAB

- The trade-off between MHA and MQA:
  - Divide query heads into g groups, each sharing a single key head and value head;
  - MHA: $g = n_H$; MQG: $g = 1$.
- Convert MHA to GQA:
  - Similar pooling as MQA

Computation	Input	Output
$Q = xW^Q$	$x \in \mathbb{R}^{L\times D}, W^Q \in \mathbb{R}^{D\times D}$	$Q \in \mathbb{R}^{L\times D}$
$K = xW^K$	$x \in \mathbb{R}^{L\times D}, W^K \in \mathbb{R}^{D\times gH}$	$K \in \mathbb{R}^{L\times gH}$
$V = xW^V$	$x \in \mathbb{R}^{L\times D}, W^V \in \mathbb{R}^{D\times gH}$	$V \in \mathbb{R}^{L\times gH}$
$[Q_1, Q_2 \dots, Q_{n_h}] = \text{Partition}_{-1}(Q)$	$Q \in \mathbb{R}^{L\times D}$	$Q_i \in \mathbb{R}^{L\times H}, i = 1, \dots n_h$
$[K_1, K_2 \dots, K_g] = \text{Partition}_{-1}(K)$	$K \in \mathbb{R}^{L\times gH}$	$K_i \in \mathbb{R}^{L\times H}, i = 1, \dots g$
$[V_1, V_2 \dots, V_g] = \text{Partition}_{-1}(V)$	$V \in \mathbb{R}^{L\times gH}$	$V_i \in \mathbb{R}^{L\times H}, i = 1, \dots g$
$\text{Score}_i = \text{softmax}(\frac{Q_i K_{\lfloor i/g \rfloor}^T}{\sqrt{D}}), i = 1, \dots n_h$	$Q_i, K_{\lfloor i/g \rfloor} \in \mathbb{R}^{L\times H}$	$\text{score}_i \in \mathbb{R}^{L\times L}$
$Z_i = \text{score}_i V_{\lfloor i/g \rfloor}, i = 1, \dots n_h$	$\text{score}_i \in \mathbb{R}^{L\times L}, V \in \mathbb{R}^{L\times H}$	$Z_i \in \mathbb{R}^{L\times H}$



Memory Footprint with MQA and GQA

RELAXED SYSTEM LAB

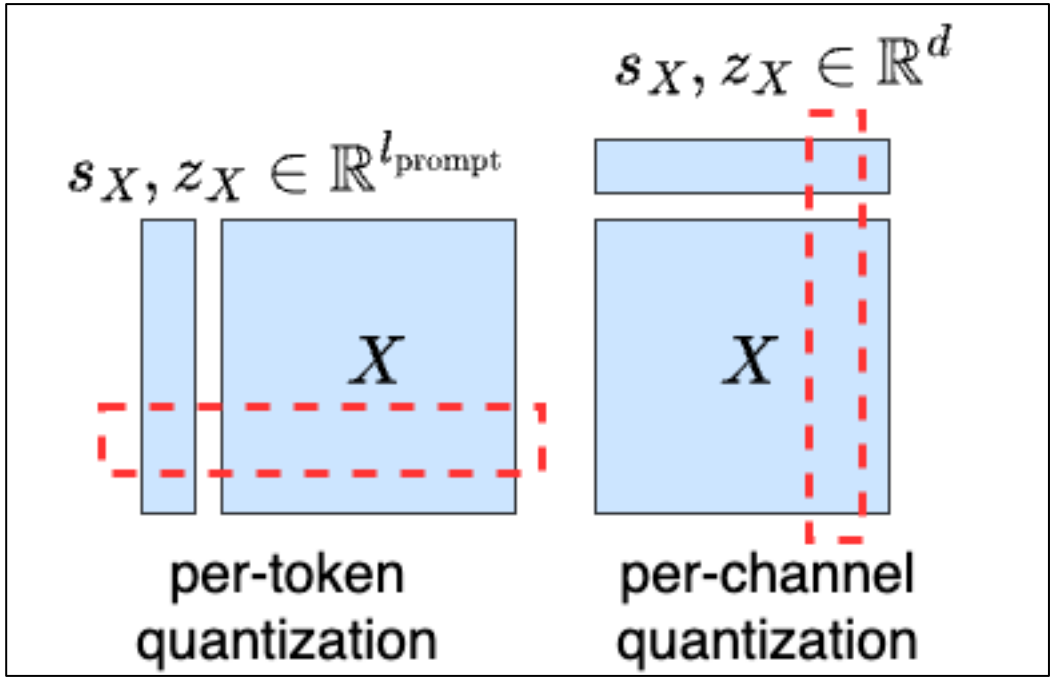
- Suppose:
  - L is the max context-length;
  - D is the model dimension;
  - Multi-head attention:
    - $D = n_H \times H$
    - H is the head dimension;
    - n_h is the number of heads.
    - g is the number of groups in GQA
  - Everything computation is in FP16

- For each transformer layer:
  - Model parameters:
    - MHA: $24D^2$
    - GQA: $2\times(D^2 + DgH + DgH + D^2 + 4D^2 + 4D^2) = 20D^2 + 4DgH$
    - MQA: $2\times(D^2 + DH + DH + D^2 + 4D^2 + 4D^2) = 20D^2 + 4DH$
  - KV cache:
    - MHA: $4LD$
    - GAQ: $2\times(LgH + LgH) = 4LgH$
    - MQA: $2\times(LH + LH) = 4LH$

KV Cache Compression – Quantization

RELAXED SYSTEM LAB

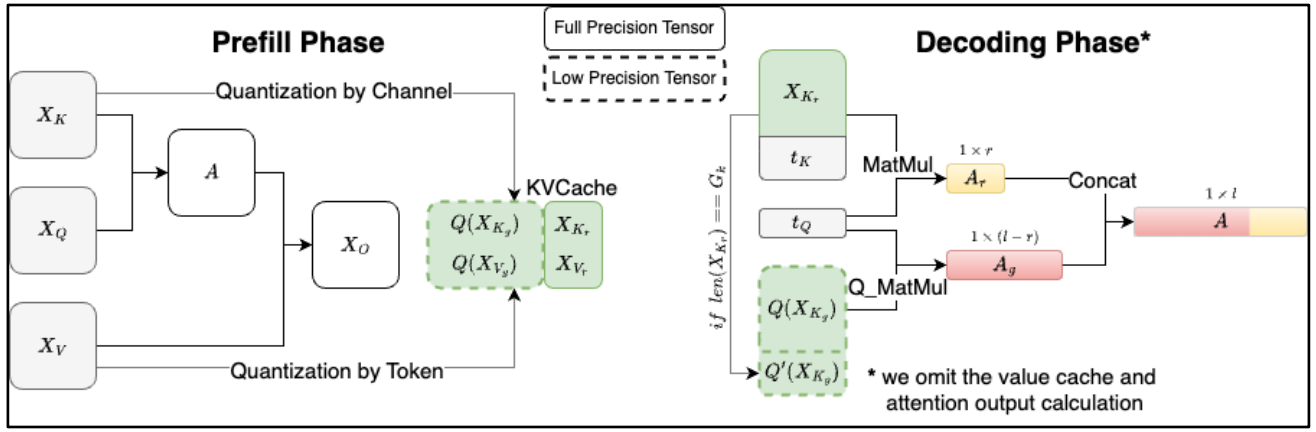
- KIVI: https://arxiv.org/abs/2402.02750
- Key observation:
  - *key cache should be quantized per-channel and value cache should be quantized per-token.*
- However, per-channel quantization, the quantization process spans across different tokens;
- Thus, this cannot be directly implemented in the streaming setting.



KV Cache Compression – Quantization

RELAXED SYSTEM LAB

- KIVI: https://arxiv.org/abs/2402.02750
- Key observation:
  - *key cache should be quantized per-channel and value cache should be quantized per-token.*
- Solution:
  - Group key cache every G tokens and quantize them separately.



Algorithm 1: The `KIVI` Prefill & Decoding Algorithm

parameter: group size G, residual length R

procedure `Prefill`:
- **Input:** $\boldsymbol{X} \in \mathbb{R}^{l_{\text{prompt}} \times d}$
- $\boldsymbol{X}_K = \boldsymbol{X}\boldsymbol{W}_K, \boldsymbol{X}_V = \boldsymbol{X}\boldsymbol{W}_V$
- $\boldsymbol{X}_{V_g} = \boldsymbol{X}_V[: l_{\text{prompt}} - R], \boldsymbol{X}_{V_r} = \boldsymbol{X}_V[l_{\text{prompt}} - R :]$
- $Q(\boldsymbol{X}_{V_g}) \leftarrow \text{GroupQuant}(\boldsymbol{X}_{V_g}, \text{dim=token}, \text{numGroup=}d//G)$
- $Q(\boldsymbol{X}_{K_g}), \boldsymbol{X}_{K_r} \leftarrow \texttt{KeyQuant}(\boldsymbol{X}_K)$
- `KV cache` $\leftarrow Q(\boldsymbol{X}_{K_g}), \boldsymbol{X}_{K_r}, Q(\boldsymbol{X}_{V_g}), \boldsymbol{X}_{V_r}$
- **return** $\boldsymbol{X}_K, \boldsymbol{X}_V$

end

procedure `Decoding`:
- **Input:** `KV cache`, $\boldsymbol{t} \in \mathbb{R}^{1 \times d}$
- $\boldsymbol{t}_Q = \boldsymbol{t}\boldsymbol{W}_Q, \boldsymbol{t}_K = \boldsymbol{t}\boldsymbol{W}_K, \boldsymbol{t}_V = \boldsymbol{t}\boldsymbol{W}_V$
- $Q(\boldsymbol{X}_{K_g}), \boldsymbol{X}_{K_r}, Q(\boldsymbol{X}_{V_g}), \boldsymbol{X}_{V_r} \leftarrow$ `KV cache`
- $\boldsymbol{X}_{K_r} \leftarrow \text{Concat}([\boldsymbol{X}_{K_r}, \boldsymbol{t}_K], \text{dim=token})$
- $\boldsymbol{X}_{V_r} \leftarrow \text{Concat}([\boldsymbol{X}_{V_r}, \boldsymbol{t}_V], \text{dim=token})$
- **if** $len(\boldsymbol{X}_{K_r}) = R$ **then**
  - $Q(\boldsymbol{X}_{K_r}), \_ \leftarrow \texttt{KeyQuant}(\boldsymbol{X}_{K_r})$
  - $Q(\boldsymbol{X}_{K_g}) \leftarrow \text{Concat}([Q(\boldsymbol{X}_{K_g}), Q(\boldsymbol{X}_{K_r})], \text{dim=token})$
  - $\boldsymbol{X}_{K_r} \leftarrow$ empty tensor.
- **end**
- **if** $len(\boldsymbol{X}_{V_r}) > R$ **then**
  - $Q(\boldsymbol{X}_{V_r'}) \leftarrow \text{GroupQuant}(\boldsymbol{X}_{V_r}[: -R], \text{dim=token}, \text{numGroup} = d//G)$
  - $Q(\boldsymbol{X}_{V_g}) \leftarrow \text{Concat}([Q(\boldsymbol{X}_{V_g}), Q(\boldsymbol{X}_{V_r'})], \text{dim=token})$
  - $\boldsymbol{X}_{V_r} \leftarrow \boldsymbol{X}_{V_r}[-R :]$
- **end**
- $\boldsymbol{A} \leftarrow \text{Concat}([\boldsymbol{t}_Q Q(\boldsymbol{X}_{K_g})^\top, \boldsymbol{t}_Q \boldsymbol{X}_{K_r}^\top], \text{dim=token})$
- $\boldsymbol{A}_g = \text{Softmax}(\boldsymbol{A})[: -R], \boldsymbol{A}_r = \text{Softmax}(\boldsymbol{A})[-R :]$
- $\boldsymbol{t}_O \leftarrow \boldsymbol{A}_g Q(\boldsymbol{X}_{V_g}) + \boldsymbol{A}_r \boldsymbol{X}_{V_r}$
- `KV cache` $\leftarrow Q(\boldsymbol{X}_{K_g}), \boldsymbol{X}_{K_r}, Q(\boldsymbol{X}_{V_g}), \boldsymbol{X}_{V_r}$
- **return** $\boldsymbol{t}_O$

end

function `KeyQuant`$(\boldsymbol{X}_K \in \mathbb{R}^{l \times d})$:
- $r = l\%R,$
- $\boldsymbol{X}_{K_g} = \boldsymbol{X}_K[: l - r], \boldsymbol{X}_{K_r} = \boldsymbol{X}_K[l - r :]$
- $Q(\boldsymbol{X}_{K_g}) \leftarrow \text{GroupQuant}(\boldsymbol{X}_{K_g}, \text{dim=channel}, \text{numGroup=}l//G)$
- **return** $Q(\boldsymbol{X}_{K_g}), \boldsymbol{X}_{K_r}$

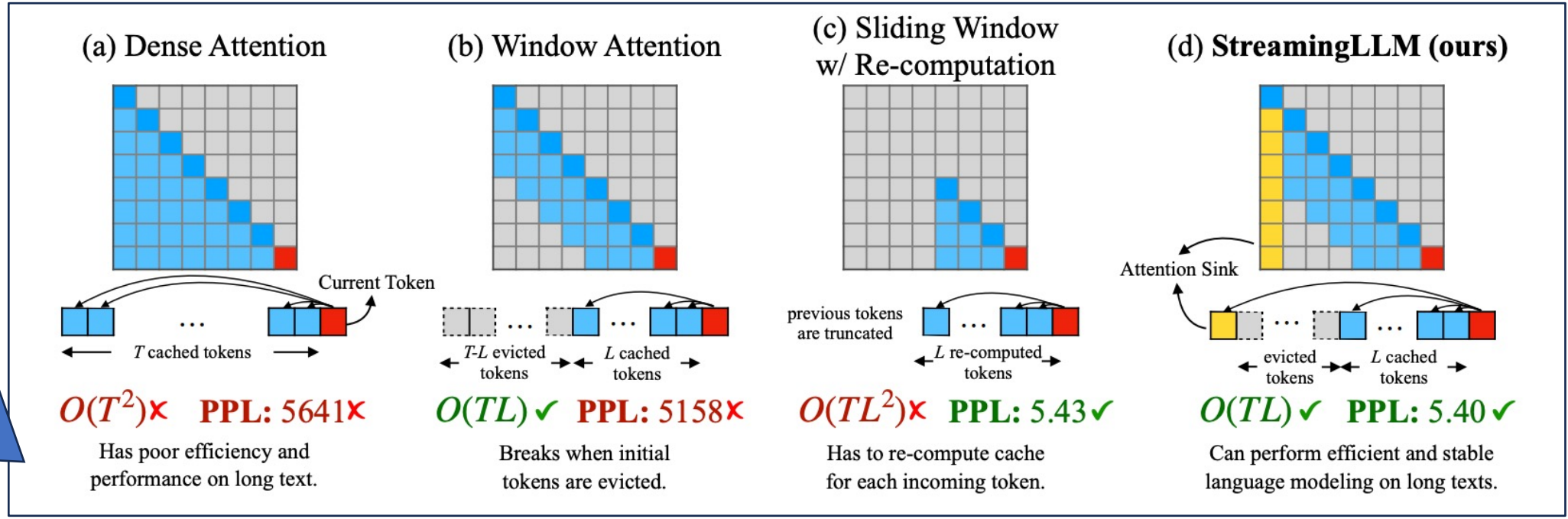
end

KV Cache Compression – Reserving More Important Tokens

RELAXED SYSTEM LAB

- StreamingLLM: https://arxiv.org/pdf/2309.17453
- Streaming based methods:

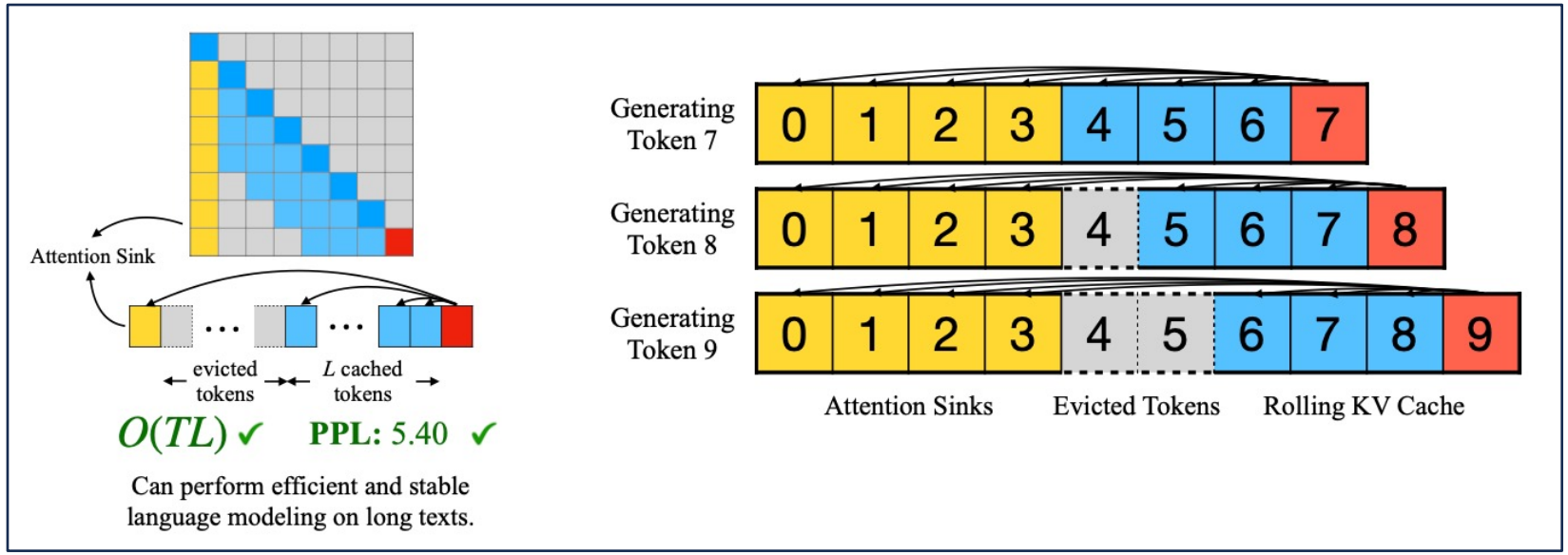
The setting here is a little different, the paper assumes the LLM is pre-trained on texts of length L, and the LLM can do generative inference with longer context T, where $T \gg L$.



KV Cache Compression – Reserving More Important Tokens

RELAXED SYSTEM LAB

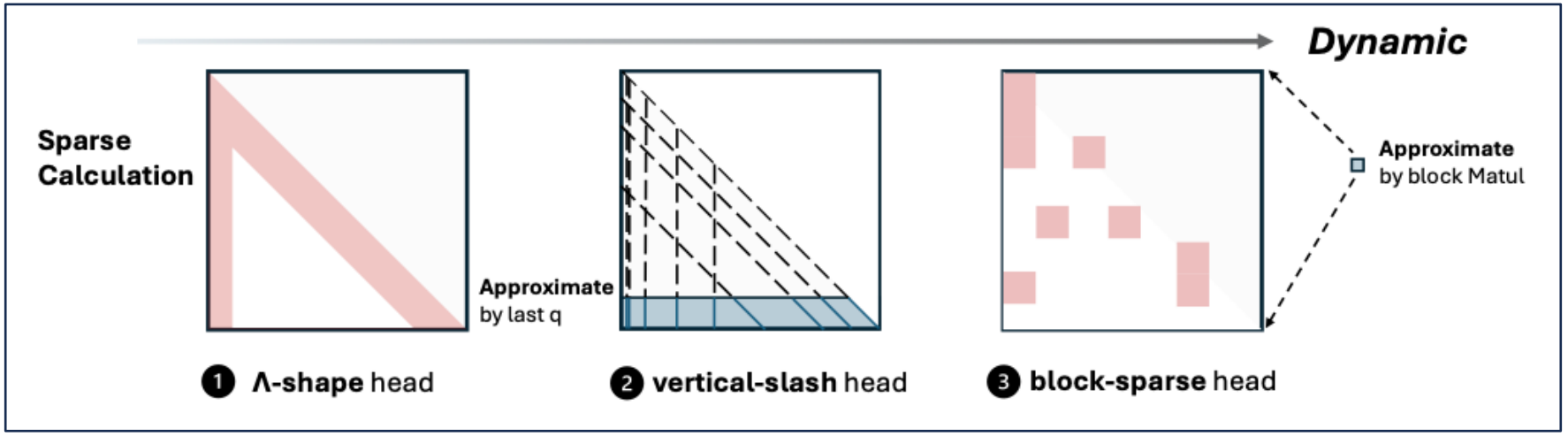
- StreamingLLM: https://arxiv.org/pdf/2309.17453
- Key idea:
  - ***Attention sink***: tokens that disproportionately attract attention irrespective of their relevance.
  - Initial tokens have large attention scores, even if they're not semantically significant.
  - Initial tokens' advantage in becoming sinks due to their visibility to subsequent tokens, rooted in autoregressive language modeling.



KV Cache Sparsity

RELAXED SYSTEM LAB

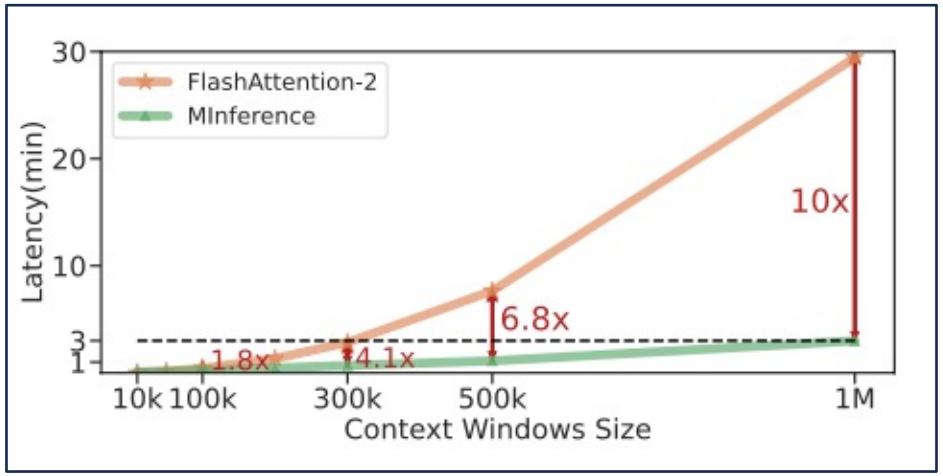
- MInference 1.0: https://arxiv.org/pdf/2407.02490
- Shared wisdom: *the attention matrices are highly sparse.*
- This is the computation bottleneck in prefill phase.
  - **A-shape pattern**: the attention weights of these types of heads are concentrated on initial tokens and local windows, exhibiting relatively higher stability.
  - **Vertical-slash (VS) pattern**: The attention weights are concentrated on specific tokens (vertical lines) and tokens at fixed intervals (slash lines).
  - **Block-sparse pattern**: This sparsity pattern is the most dynamic, exhibiting a more dispersed distribution.



KV Cache Sparsity

RELAXED SYSTEM LAB

- MInference 1.0: https://arxiv.org/pdf/2407.02490
- Summary of the solution:
  - Offline attention pattern identification for each head;
  - Dynamic build of sparse indices w.r.t. the pattern;
  - Sparse attention calculation with optimized GPU kernels.
- Open source implementation: https://github.com/microsoft/MInference/blob/main/minference/modules/minference_forward.py



References

- https://towardsdatascience.com/introduction-to-weight-quantization-2494701b9c0c
- https://arxiv.org/abs/2210.17323
- https://huggingface.co/blog/gptq-integration
- https://arxiv.org/abs/2402.13116
- https://arxiv.org/abs/1911.02150
- https://arxiv.org/abs/2305.13245
- https://arxiv.org/abs/2402.02750
- https://arxiv.org/abs/2309.17453
- https://arxiv.org/abs/2306.14048
- https://arxiv.org/pdf/2407.02490