



Denglin HammingTM V2

dIDNN Introduction

DL-DG/SW-036A-03

2025-02-27

Copyright©苏州登临科技有限公司，2019 - 2025，版权所有。

未经苏州登临科技有限公司事先书面同意，不得以任何形式或方式复制或传播本文件的任何部分。

商标和许可



和其它苏州登临科技有限公司的其它登临科技的图标为苏州登临科技有限公司的商标。本手册中提及的所有其他商标均为其各自所有者的财产。

通知

所购买的产品、服务和特性由苏州登临科技有限公司与客户签订的合同规定。本文件中描述的所有或部分产品、服务和特性可能不在采购范围或使用范围内。除非合同中另有规定，本文件中的所有声明、信息和建议均按“原样”提供，无任何明示或暗示的保证或陈述。

本手册中的信息如有更改，恕不另行通知。本文件在编制过程中已尽一切努力确保内容的准确性，本文件中的所有声明、信息和建议不构成任何明示或暗示的保证。

苏州登临科技有限公司

苏州工业园区扬富路11号南岸新地一期商务楼5号1101室，江苏，中国

<http://www.denglin.ai>

Email: support@denglin.ai

更新历史

版本	更新描述
03	新增API描述及内容优化。
02	内容优化。
01	第一次发布。

章节目录

- 1 简介
- 2 版本
- 3 dIDNN 算子
 - 3.1 cuDNN 算子
 - 3.2 dIDNN 扩展算子
- 4 dIDNN 扩展算子相关 API 接口描述
 - 4.1 cudnnScaledDotProductAttention
 - 4.1.1 cudnnScaledDotProductAttention
 - 4.1.2 cudnnGetScaledDotProductAttentionWorkspaceSize
 - 4.2 cudnnScaledDotProductAttentionEx [placeholder]
 - 4.3 cudnnScaledDotProductAttentionBackward [placeholder]
 - 4.4 cudnnMHAForward
 - 4.4.1 cudnnMHAForward
 - 4.4.2 cudnnGetMHAForwardWorkspaceSize
 - 4.5 cudnnMHABackward [placeholder]
 - 4.6 cudnnMHAVarlenForward
 - 4.6.1 cudnnMHAVarlenForward
 - 4.6.2 cudnnGetMHAVarlenForwardWorkspaceSize
 - 4.7 cudnnMHAVarlenBackward [placeholder]
 - 4.8 cudnnMHAForwardKVCache
 - 4.8.1 cudnnMHAForwardKVCache
 - 4.8.2 cudnnGetMHAForwardKVCacheWorkspaceSize
 - 4.9 cudnnMaxPoolingIndicesBackward [placeholder]
 - 4.10 cudnnMaxPoolingIndicesForward [placeholder]
 - 4.11 cudnnFusedMoeForwardInference [placeholder]

1 简介

dIDNN是加速深度神经网络算子计算的软件库，支持神经网络算子在登临人工智能处理器上高效地运算。

2 版本

dIDNN支持部分cuDNN原生算子及dIDNN扩展算子。

- 1. cuDNN原生算子支持对应cuDNN **8.5.0**。
- 2. dIDNN扩展算子支持当前版本为 **1.6.0**。

3 dIDNN 算子

dIDNN 软件库支持cuDNN算子和dIDNN扩展算子实现。

3.1 cuDNN 算子

cuDNN算子支持列表如下：

序号	cuDNN 算子
1	cudaActivationBackward
2	cudaActivationForward
3	cudaAddTensor
4	cudaBatchNormalizationBackward
5	cudaBatchNormalizationBackwardEx
6	cudaBatchNormalizationForwardInference
7	cudaBatchNormalizationForwardTraining
8	cudaBatchNormalizationForwardTrainingEx
9	cudaConvolutionBackwardBias
10	cudaConvolutionBackwardData
11	cudaConvolutionBackwardFilter
12	cudaConvolutionForward
13	cudaCTCLoss
14	cudaDropoutBackward

序号	cuDNN 算子
15	cudaDnnDropoutForward
16	cudaDnnIm2Col
17	cudaDnnOpTensor
18	cudaDnnPoolingBackward
19	cudaDnnPoolingForward
20	cudaDnnReduceTensor
21	cudaDnnSoftmaxBackward
22	cudaDnnSoftmaxForward
23	cudaDnnTransformTensor

API文档参考：[cuDNN8.5.0 API Reference](#)

3.2 dIDNN 扩展算子

dIDNN 扩展算子列表如下：

序号	dIDNN 扩展算子	说明
1	cudaDnnScaledDotProductAttention	-
2	cudaDnnScaledDotProductAttentionEx	[placeholder]
3	cudaDnnScaledDotProductAttentionBackward	[placeholder]
4	cudaDnnMHAFoward	-
5	cudaDnnMHABackward	[placeholder]
6	cudaDnnMHAVarlenForward	-
7	cudaDnnMHAVarlenBackward	[placeholder]
8	cudaDnnMHAFowardKVCache	-
9	cudaDnnMaxPoolingIndicesBackward	[placeholder]
10	cudaDnnMaxPoolingIndicesForward	[placeholder]
11	cudaDnnFusedMoeForwardInference	[placeholder]

4 dIDNN 扩展算子相关 API 接口描述

本章描述了dIDNN所扩展的自定义算子APIs及相关的辅助APIs。

4.1 cudnnScaledDotProductAttention

4.1.1 cudnnScaledDotProductAttention

```

cudnnStatus_t CUDNNWINAPI cudnnScaledDotProductAttention(
    cudnnHandle_t handle,
    const cudnnTensorDescriptor_t q_desc,
    const void *q,
    const cudnnTensorDescriptor_t k_desc,
    const void *k,
    const cudnnTensorDescriptor_t v_desc,
    const void *v,
    const cudnnTensorDescriptor_t mask_desc,
    const void *mask,
    float dropout,
    bool is_causal,
    float scale,
    void *workspace,
    size_t workspace_size_in_bytes,
    const cudnnTensorDescriptor_t out_desc,
    void *out);

```

This function computes scaled dot product attention on query, key and value tensors, using an optional attention mask if passed, and applying dropout if a probability greater than 0.0 is specified.

参数

- **handle**
Input. Handle to a previously created cuDNN context. For more information, refer to `cudnnHandle_t`.
- **q_desc**
Input. Handle to a previously initialized query tensor descriptor. For more information, refer to `cudnnTensorDescriptor_t`.
- **q**
Input. Data pointer to GPU memory associated with the query tensor descriptor `q_desc`.
- **k_desc**
Input. Handle to a previously initialized key tensor descriptor. For more information, refer to `cudnnTensorDescriptor_t`.
- **k**
Input. Data pointer to GPU memory associated with the key tensor descriptor `k_desc`.

- **v_desc**

Input. Handle to a previously initialized value tensor descriptor. For more information, refer to `cudaTensorDescriptor_t`.

- **v**

Input. Data pointer to GPU memory associated with the value tensor descriptor `v_desc`.

- **mask_desc**

Input. Handle to a previously initialized mask tensor descriptor. For more information, refer to `cudaTensorDescriptor_t`.

If mask is not applied, this argument can be NULL.

- **mask**

Input. Data pointer to GPU memory associated with the mask tensor descriptor `mask_desc`.

If mask is not applied, this argument can be NULL.

- **dropout**

Input. Dropout probability; if greater than 0.0, dropout is applied.

- **is_causal**

Input. If true, assumes upper left causal attention masking and errors if both `attn_mask` and `is_causal` are set.

- **scale**

Input. Scaling factor applied prior to softmax.

- **workspace**

Input. Data pointer to GPU memory to a workspace needed to be able to execute the specified algorithm.

If no workspace is needed for a particular algorithm, that pointer can be nil.

- **workspace_size_in_bytes**

Input. Specifies the size in bytes of the provided workspace.

- **out_desc**

Input. Handle to a previously initialized attention output tensor descriptor.

- **out**

Output. Data pointer to GPU memory associated with the attention output tensor descriptor `out_desc` that carries the result.

说明

1. Currently, only 4D dimension tensors of q, k, v, mask, out are supported.
2. By convention, dimension order of q, k, v, mask, out tensor descriptor aligns with pytorch `scaled_dot_product_attention` API: `[batch, head_num, seq_len, head_size]`.
3. `mask_desc` & `mask` can be NULL or not NULL at the same time.
4. When `is_causal` is set, `mask_desc` & `mask` has both to be NULL.

返回值

- **CUDNN_STATUS_SUCCESS**

The operation was launched successfully.

- **CUDNN_STATUS_BAD_PARAM**

At least one of the following conditions is met:

- At least one of the following params is NULL: handle, q_desc, k_desc, v_desc, out_desc, q, k, v, out.
- Number of dimensions of q_desc, k_desc, v_desc, out_desc, mask_desc(if not NULL) is not 4.
- Dimension sizes of tensor q, k, v, mask, out do not match with rule described in https://pytorch.org/docs/stable/generated/torch.nn.functional.scaled_dot_product_attention.html.
- Data types of tensor q, k, v, out are not the same.
- Data types of tensor q, k, v, out are not CUDNN_DATA_HALF or CUDNN_DATA_FLOAT.
- Data types of tensor mask is not CUDNN_DATA_INT8 or CUDNN_DATA_UINT8 or not matching with tensor q.
- mask_desc & mask are not both NULL or both not NULL.
- When is_causal is set, mask_desc or mask is not NULL.
- workspace_size_in_bytes is 0 while workspace is not null.

- **CUDNN_STATUS_EXECUTION_FAILED**

The function failed to launch on the GPU.

4.1.2 cudnnGetScaledDotProductAttentionWorkspaceSize

```

cudnnStatus_t CUDNNWINAPI cudnnGetScaledDotProductAttentionWorkspaceSize(
    cudnnHandle_t handle,
    const cudnnTensorDescriptor_t q_desc,
    const cudnnTensorDescriptor_t k_desc,
    const cudnnTensorDescriptor_t v_desc,
    const cudnnTensorDescriptor_t mask_desc,
    const cudnnTensorDescriptor_t out_desc,
    float dropout,
    bool is_causal,
    float scale,
    size_t *size_in_bytes);

```

This function returns the amount of GPU memory workspace the user needs to allocate to be able to call `cudnnScaledDotProductAttention()`.

参数

- **handle**

Input. Handle to a previously created cuDNN context. For more information, refer to `cudnnHandle_t`.

- **q_desc**

Input. Handle to a previously initialized query tensor descriptor. For more information, refer to `cudnnTensorDescriptor_t`.

- **k_desc**

Input. Handle to a previously initialized key tensor descriptor. For more information, refer to `cudnnTensorDescriptor_t`.

- **v_desc**

Input. Handle to a previously initialized value tensor descriptor. For more information, refer to `cudaTensorDescriptor_t`.

- **mask_desc**

Input. Handle to a previously initialized mask tensor descriptor. For more information, refer to `cudaTensorDescriptor_t`.

If mask is not applied, this argument can be NULL.

- **out_desc**

Input. Handle to a previously initialized attention output tensor descriptor.

- **dropout**

Input. Dropout probability; if greater than 0.0, dropout is applied.

- **is_causal**

Input. If true, assumes upper left causal attention masking and errors if both `attn_mask` and `is_causal` are set.

- **scale**

Input. Scaling factor applied prior to softmax.

- **size_in_bytes**

Output. Amount of GPU memory needed as workspace to be able to execute `cudaScaledDotProductAttention()`.

说明

1. Currently, only 4D dimension tensors of q, k, v, mask, out are supported.
2. By convention, dimension order of q, k, v, mask, out tensor descriptor aligns with pytorch `scaled_dot_product_attention` API: [batch, head_num, seq_len, head_size].
3. When `is_causal` is set, `mask_desc` has to be NULL.

返回值

- **CUDNN_STATUS_SUCCESS**

The operation was launched successfully.

- **CUDNN_STATUS_BAD_PARAM**

At least one of the following conditions is met:

- At least one of the following params is NULL: `handle`, `q_desc`, `k_desc`, `v_desc`, `out_desc`, `size_in_bytes`.
- Number of dimensions of `q_desc`, `k_desc`, `v_desc`, `out_desc`, `mask_desc`(if not NULL) is not 4.
- Dimension sizes of tensor q, k, v, mask, out do not match with rule described in https://pytorch.org/docs/stable/generated/torch.nn.functional.scaled_dot_product_attention.html.
- Data types of tensor q, k, v, out are not the same.
- Data types of tensor q, k, v, out are not `CUDNN_DATA_HALF` or `CUDNN_DATA_FLOAT`.
- Data types of tensor mask is not `CUDNN_DATA_INT8` or `CUDNN_DATA_UINT8` or not matching with tensor q.
- When `is_causal` is set, `mask_desc` is not NULL.

- **CUDNN_STATUS_EXECUTION_FAILED**

The function failed to launch on the GPU.

4.2 cudnnScaledDotProductAttentionEx [placeholder]

4.3 cudnnScaledDotProductAttentionBackward [placeholder]

4.4 cudnnMHAForward

4.4.1 cudnnMHAForward

```

cudnnStatus_t CUDNNWINAPI cudnnMHAForward(
    cudnnHandle_t handle,
    const cudnnTensorDescriptor_t q_desc,
    const void *q,
    const cudnnTensorDescriptor_t k_desc,
    const void *k,
    const cudnnTensorDescriptor_t v_desc,
    const void *v,
    const cudnnTensorDescriptor_t alibi_slopes_desc,
    void *alibi_slopes,
    const cudnnTensorDescriptor_t out_desc,
    void *out,
    const cudnnTensorDescriptor_t softmax_lse_desc,
    void *softmax_lse,
    const cudnnTensorDescriptor_t p_desc,
    void *p,
    float dropout,
    float softmax_scale,
    bool is_causal,
    int window_size_left,
    int window_size_right,
    bool return_softmax,
    unsigned long long *philox_seed,
    unsigned long long *philox_offset,
    void *workspace,
    size_t workspace_size_in_bytes);

```

This function supports flash attention computation defined by repo [flash-attention](#).

参数

- **handle**

Input. Handle to a previously created cuDNN context. For more information, refer to `cudnnHandle_t`.

- **q_desc**

Input. Handle to a previously initialized query tensor descriptor. For more information, refer to `cudnnTensorDescriptor_t`.

- **q**

Input. Data pointer to GPU memory associated with the query tensor descriptor `q_desc`.

- **k_desc**

Input. Handle to a previously initialized key tensor descriptor. For more information, refer to `cudaTensorDescriptor_t`.

- **k**

Input. Data pointer to GPU memory associated with the key tensor descriptor `k_desc`.

- **v_desc**

Input. Handle to a previously initialized value tensor descriptor. For more information, refer to `cudaTensorDescriptor_t`.

- **v**

Input. Data pointer to GPU memory associated with the value tensor descriptor `v_desc`.

- **alibi_slopes_desc**

Input. Handle to a previously initialized alibi slopes tensor descriptor. For more information, refer to `cudaTensorDescriptor_t`.

If alibi slopes is not applied, this argument can be NULL.

- **alibi_slopes**

Input. Data pointer to GPU memory associated with the alibi slopes tensor descriptor `alibi_slopes_desc`.

A bias of $(-alibi_slope * |i + seqlen_k - seqlen_q - j|)$ is added to the attention score of query i and key j .

If alibi slopes is not applied, this argument can be NULL.

- **out_desc**

Input. Handle to a previously initialized attention output tensor descriptor.

- **out**

Output. Data pointer to GPU memory associated with the attention output tensor descriptor `out_desc` that carries the result.

- **softmax_lse_desc**

Input. Handle to a previously initialized softmax_lse tensor descriptor.

If `return_softmax` is false, this argument can be NULL.

- **softmax_lse**

Output. Data pointer to GPU memory associated with the softmax_lse tensor descriptor `softmax_lse_desc`.

The logsumexp of each row of the matrix $QK^T * scaling$ (e.g., log of the softmax normalization factor).

If `return_softmax` is false, this argument can be NULL.

- **p_desc**

Input. Handle to a previously initialized p tensor descriptor. For more information, refer to `cudaTensorDescriptor_t`.

If `return_softmax` is false, this argument can be NULL.

- **p**

Output. Data pointer to GPU memory associated with the *p* tensor descriptor *p_desc*.
 The output of softmax (possibly with different scaling). It also encodes the dropout pattern (negative means that location was dropped, nonnegative means it was kept).
 If *return_softmax* is false, this argument can be NULL.

- **dropout**

Input. Dropout probability; if greater than 0.0, dropout is applied.

- **softmax_scale**

Input. The scaling of QK^T before applying softmax. Default to $1 / \sqrt{\text{headdim}}$.

- **is_causal**

Input. Whether to apply causal attention mask.

- **window_size_left**

Input. Implements sliding window local attention with *window_size_right* together.

- **window_size_right**

Input. Implements sliding window local attention with *window_size_left* together.

- **return_softmax**

Input. Whether to return tensor *softmax_lse* and *p*.

- **philox_seed**

Output. *philox_seed* and *philox_offset* together defines the *rng_state* for dropout.

- **philox_offset**

Output. *philox_seed* and *philox_offset* together defines the *rng_state* for dropout.

- **workspace**

Input. Data pointer to GPU memory to a workspace needed to be able to execute the specified algorithm.
 If no workspace is needed for a particular algorithm, that pointer can be nil.

- **workspace_size_in_bytes**

Input. Specifies the size in bytes of the provided workspace.

说明

1. Supports multi-query and grouped-query attention (MQA/GQA) by passing in KV with fewer heads than Q. Note that the number of heads in Q **must be divisible** by the number of heads in KV.
2. If *is_causal*=True, the causal mask is aligned to the bottom right corner of the attention matrix.
3. If *window_size* != (-1, -1), implements sliding window local attention. Query at position *i* will only attend to keys between $[i + \text{seqlen_k} - \text{seqlen_q} - \text{window_size}[0], i + \text{seqlen_k} - \text{seqlen_q} + \text{window_size}[1]]$ inclusive.
4. Only 4D dimension tensors of *q*, *k*, *v*, *out* are supported, and in dims of: $[\text{batch_size}, \text{seqlen_q}/\text{seqlen_kv}, \text{nheads_q}/\text{nheads_kv}, \text{headdim}]$.
5. Tensor *alibi_slopes* should be in dims of $[1, 1, \text{nheads_q}]$ or $[1, \text{batch_size}, \text{nheads_q}]$ and data type of CUDNN_DATA_FLOAT.
6. Tensor *softmax_lse* should be in dims of $[\text{batch_size}, \text{nheads_q}, \text{seqlen_q}]$ and data type of CUDNN_DATA_FLOAT.

7. Tensor p is the same as the tensor S_dmask described in [flash attention repo interface](#), which is in dims of [batch_size, nheads_q, seqlen_q, seqlen_kv].
8. Data types of tensor q, k, v, out, p are only supported with CUDNN_DATA_HALF, CUDNN_DATA_FLOAT, CUDNN_DATA_BFLOAT16.

返回值

- **CUDNN_STATUS_SUCCESS**

The operation was launched successfully.

- **CUDNN_STATUS_BAD_PARAM**

At least one of the following conditions is met:

- At least one of the following params is NULL: *handle, q_desc, k_desc, v_desc, out_desc, q, k, v, out*.
- Data types of tensor q, k, v, out are not the same.
- When *return_softmax* is TRUE, *softmax_lse_desc, softmax_lse, p_desc, p* are NULL.
- *alibi_slopes_desc* & *alibi_slopes* are not both NULL or both not NULL.
- *softmax_lse_desc* & *softmax_lse* are not both NULL or both not NULL.
- *p_desc* & *p* are not both NULL or both not NULL.
- *workspace_size_in_bytes* is 0 while *workspace* is not NULL.
- Rules mentioned above in parameter descriptions are violated.

- **CUDNN_STATUS_EXECUTION_FAILED**

The function failed to launch on the GPU.

4.4.2 cudnnGetMHAForwardWorkspaceSize

```

cudnnStatus_t CUDNNWINAPI cudnnGetMHAForwardWorkspaceSize(
    cudnnHandle_t handle,
    const cudnnTensorDescriptor_t q_desc,
    const cudnnTensorDescriptor_t k_desc,
    const cudnnTensorDescriptor_t v_desc,
    const cudnnTensorDescriptor_t alibi_slopes_desc,
    const cudnnTensorDescriptor_t out_desc,
    const cudnnTensorDescriptor_t softmax_lse_desc,
    const cudnnTensorDescriptor_t p_desc,
    float dropout,
    float softmax_scale,
    bool is_causal,
    int window_size_left,
    int window_size_right,
    bool return_softmax,
    size_t *size_in_bytes);

```

This function returns the amount of GPU memory workspace the user needs to allocate to be able to call `cudnnMHAForward()`.

参数

- **handle**

Input. Handle to a previously created cuDNN context. For more information, refer to `cudaDnnHandle_t`.

- **q_desc**

Input. Handle to a previously initialized query tensor descriptor. For more information, refer to `cudaDnnTensorDescriptor_t`.

- **k_desc**

Input. Handle to a previously initialized key tensor descriptor. For more information, refer to `cudaDnnTensorDescriptor_t`.

- **v_desc**

Input. Handle to a previously initialized value tensor descriptor. For more information, refer to `cudaDnnTensorDescriptor_t`.

- **alibi_slopes_desc**

Input. Handle to a previously initialized alibi slopes tensor descriptor. For more information, refer to `cudaDnnTensorDescriptor_t`.

If alibi slopes is not applied, this argument can be NULL.

- **out_desc**

Input. Handle to a previously initialized attention output tensor descriptor.

- **softmax_lse_desc**

Input. Handle to a previously initialized softmax_lse tensor descriptor.

If `return_softmax` is false, this argument can be NULL.

- **p_desc**

Input. Handle to a previously initialized p tensor descriptor. For more information, refer to `cudaDnnTensorDescriptor_t`.

If `return_softmax` is false, this argument can be NULL.

- **dropout**

Input. Dropout probability; if greater than 0.0, dropout is applied.

- **softmax_scale**

Input. The scaling of QK^T before applying softmax. Default to $1 / \sqrt{\text{headdim}}$.

- **is_causal**

Input. Whether to apply causal attention mask.

- **window_size_left**

Input. Implements sliding window local attention with `window_size_right` together.

- **window_size_right**

Input. Implements sliding window local attention with `window_size_left` together.

- **return_softmax**

Input. Whether to return tensor `softmax_lse` and `p`.

- **size_in_bytes**

Output. Amount of GPU memory needed as workspace to be able to execute `cudaDnnMHAFoward()`.

说明

1. Only 4D dimension tensors of q , k , v , out are supported, and in dims of: [batch_size, seqlen_q/seqlen_kv, nheads_q/nheads_kv, headdim]
2. Tensor $alibi_slopes$ should be in dims of [1, 1, nheads_q] or [1, batch_size, nheads_q] and data type of CUDNN_DATA_FLOAT.
3. Tensor $softmax_lse$ should be in dims of [batch_size, nheads_q, seqlen_q] and data type of CUDNN_DATA_FLOAT.
4. Tensor p is the same as the tensor S_dmask described in [flash attention repo interface](#), which is in dims of [batch_size, nheads_q, seqlen_q, seqlen_kv].
5. Data types of tensor q , k , v , out , p are only supported with CUDNN_DATA_HALF, CUDNN_DATA_FLOAT, CUDNN_DATA_BFLOAT16.

返回值

- **CUDNN_STATUS_SUCCESS**

The operation was launched successfully.

- **CUDNN_STATUS_BAD_PARAM**

At least one of the following conditions is met:

- At least one of the following params is NULL: $handle$, q_desc , k_desc , v_desc , out_desc , $size_in_bytes$.
- Data types of tensor q , k , v , out are not the same.
- When $return_softmax$ is TRUE, $softmax_lse_desc$, p_desc are NULL.
- Rules mentioned above in parameter descriptions are violated.

- **CUDNN_STATUS_EXECUTION_FAILED**

The function failed to launch on the GPU.

4.5 cudnnMHABackward [placeholder]

4.6 cudnnMHAVarlenForward

4.6.1 cudnnMHAVarlenForward

```
cudnnStatus_t CUDNNWINAPI cudnnMHAVarlenForward(
    cudnnHandle_t handle,
    const cudnnTensorDescriptor_t q_desc,
    const void *q,
    const cudnnTensorDescriptor_t k_desc,
    const void *k,
    const cudnnTensorDescriptor_t v_desc,
    const void *v,
    const cudnnTensorDescriptor_t cu_seqLens_q_desc,
    const void *cu_seqLens_q,
    const cudnnTensorDescriptor_t cu_seqLens_k_desc,
```



```

const void *cu_seqlens_k,
const cudnnTensorDescriptor_t sequested_k_desc,
const void *sequested_k,
const cudnnTensorDescriptor_t block_table_desc,
const void *block_table,
const cudnnTensorDescriptor_t alibi_slopes_desc,
void *alibi_slopes,
const cudnnTensorDescriptor_t out_desc,
void *out,
const cudnnTensorDescriptor_t softmax_lse_desc,
void *softmax_lse,
const cudnnTensorDescriptor_t p_desc,
void *p,
int max_seq_len_q,
int max_seq_len_k,
float dropout,
float softmax_scale,
bool zero_tensors,
bool is_causal,
int window_size_left,
int window_size_right,
bool return_softmax,
unsigned long long *philox_seed,
unsigned long long *philox_offset,
void *workspace,
size_t workspace_size_in_bytes);

```

This function supports varlen flash attention computation defined by repo [flash-attention](#).

参数

- **handle**

Input. Handle to a previously created cuDNN context. For more information, refer to `cudnnHandle_t`.

- **q_desc**

Input. Handle to a previously initialized query tensor descriptor. For more information, refer to `cudnnTensorDescriptor_t`.

- **q**

Input. Data pointer to GPU memory associated with the query tensor descriptor `q_desc`.

- **k_desc**

Input. Handle to a previously initialized key tensor descriptor. For more information, refer to `cudnnTensorDescriptor_t`.

- **k**

Input. Data pointer to GPU memory associated with the key tensor descriptor `k_desc`.

- **v_desc**

Input. Handle to a previously initialized value tensor descriptor. For more information, refer to `cudnnTensorDescriptor_t`.

- **v**

Input. Data pointer to GPU memory associated with the value tensor descriptor `v_desc`.

- **cu_seqlens_q_desc**

Input. Handle to a previously initialized `cu_seqlens_q` tensor descriptor. For more information, refer to `cudaTensorDescriptor_t`.

- **cu_seqlens_q**

Input. Data pointer to GPU memory associated with the `cu_seqlens_q` tensor descriptor `cu_seqlens_q_desc`. The cumulative sequence lengths of the sequences in the batch, used to index into `q`.

- **cu_seqlens_k_desc**

Input. Handle to a previously initialized `cu_seqlens_k` tensor descriptor. For more information, refer to `cudaTensorDescriptor_t`.

- **cu_seqlens_k**

Input. Data pointer to GPU memory associated with the `cu_seqlens_k` tensor descriptor `cu_seqlens_k_desc`. The cumulative sequence lengths of the sequences in the batch, used to index into `kv`.

- **sequested_k_desc**

Input. Handle to a previously initialized `sequested_k` tensor descriptor. For more information, refer to `cudaTensorDescriptor_t`.

This argument is optional and should be set as `NULL` if not applied.

- **sequested_k**

Input. Data pointer to GPU memory associated with the `sequested_k` tensor descriptor `sequested_k_desc`. This argument is optional and should be set as `NULL` if not applied.

- **block_table_desc**

Input. Handle to a previously initialized `block_table` tensor descriptor. For more information, refer to `cudaTensorDescriptor_t`.

This argument is optional and should be set as `NULL` if not applied.

- **block_table**

Input. Data pointer to GPU memory associated with the `block_table` tensor descriptor `block_table_desc`. This argument is optional and should be set as `NULL` if not applied.

- **alibi_slopes_desc**

Input. Handle to a previously initialized alibi slopes tensor descriptor. For more information, refer to `cudaTensorDescriptor_t`.

If alibi slopes is not applied, this argument can be `NULL`.

- **alibi_slopes**

Input. Data pointer to GPU memory associated with the alibi slopes tensor descriptor `alibi_slopes_desc`. A bias of $(-alibi_slope * |i + seqlen_k - seqlen_q - j|)$ is added to the attention score of query `i` and key `j`. If alibi slopes is not applied, this argument can be `NULL`.

- **out_desc**

Input. Handle to a previously initialized attention output tensor descriptor.

- **out**

Output. Data pointer to GPU memory associated with the attention output tensor descriptor `out_desc` that carries the result.

- **softmax_lse_desc**

Input. Handle to a previously initialized `softmax_lse` tensor descriptor.

If `return_softmax` is false, this argument can be NULL.

- **softmax_lse**

Output. Data pointer to GPU memory associated with the `softmax_lse` tensor descriptor `alibi_slopes_desc`.

The logsumexp of each row of the matrix $QK^T * \text{scaling}$ (e.g., log of the softmax normalization factor).

If `return_softmax` is false, this argument can be NULL.

- **p_desc**

Input. Handle to a previously initialized `p` tensor descriptor. For more information, refer to `cudaTensorDescriptor_t`.

If `return_softmax` is false, this argument can be NULL.

- **p**

Output. Data pointer to GPU memory associated with the `p` tensor descriptor `p_desc`.

The output of softmax (possibly with different scaling). It also encodes the dropout pattern (negative means that location was dropped, nonnegative means it was kept).

If `return_softmax` is false, this argument can be NULL.

- **max_seqlen_q**

Input. Maximum query sequence length in the batch.

- **max_seqlen_k**

Input. Maximum key sequence length in the batch.

- **dropout**

Input. Dropout probability; if greater than 0.0, dropout is applied.

- **softmax_scale**

Input. The scaling of QK^T before applying softmax. Default to $1 / \sqrt{\text{headdim}}$.

- **zero_tensors**

Input. Reserved, currently not used.

- **is_causal**

Input. Whether to apply causal attention mask.

- **window_size_left**

Input. Implements sliding window local attention with `window_size_right` together.

- **window_size_right**

Input. Implements sliding window local attention with `window_size_left` together.

- **return_softmax**

Input. Whether to return tensor `softmax_lse` and `p`. Return softmax is only supported when dropout is applied (greater than 0.0).

- **philox_seed**

Output. `philox_seed` and `philox_offset` together defines the `rng_state` for dropout.

- **philox_offset**

Output. philox_seed and philox_offset together defines the *rng_state* for dropout.

- **workspace**

Input. Data pointer to GPU memory to a workspace needed to be able to execute the specified algorithm.

If no workspace is needed for a particular algorithm, that pointer can be nil.

- **workspace_size_in_bytes**

Input. Specifies the size in bytes of the provided workspace.

说明

1. Supports multi-query and grouped-query attention (MQA/GQA) by passing in KV with fewer heads than Q. Note that the number of heads in Q **must be divisible** by the number of heads in KV.
2. If *is_causal*=True, the causal mask is aligned to the bottom right corner of the attention matrix.
3. If *window_size* != (-1, -1), implements sliding window local attention. Query at position *i* will only attend to keys between
[*i* + *seqlen_k* - *seqlen_q* - *window_size*[0], *i* + *seqlen_k* - *seqlen_q* + *window_size*[1]] inclusive.
4. Tensor *q* and *out* should be in dims of [*total_q*, *nheads_q*, *headdim*], where *total_q* = total number of query tokens in the batch.
5. If *block_table* is not applied, tensor *k* and tensor *v* should be in dims of [*total_k*, *nheads_k*, *headdim*], where *total_k* = total number of key tokens in the batch.
Otherwise, they should be in dims of [*num_blocks*, *page_block_size*, *nheads_k*, *headdim*].
6. Tensor *cu_seqlens_q* & *cu_seqlens_k* should be in dims of [1, 1, *batch_size*+1] (cuDNN asks for at least 3 dims for tensor).
7. Tensor *seqlen_k* should be in dims of [1, 1, *batch_size*].
8. Tensor *block_table* should be in dims of [1, *batch_size*, *max_num_blocks_per_seq*].
9. Tensor *alibi_slopes* should be in dims of [1, 1, *nheads_q*] or [1, *batch_size*, *nheads_q*] and data type of CUDNN_DATA_FLOAT.
10. Tensor *softmax_lse* should be in dims of [1, *nheads_q*, *total_q*] and data type of CUDNN_DATA_FLOAT.
11. Tensor *p* is the same as the tensor *S_dmask* described in [flash attention repo interface](#), which is in dims of [*batch_size*, *nheads_q*, *seqlen_q_rounded*, *seqlen_kv_rounded*].
seqlen_q_rounded/*seqlen_kv_rounded* are the round multiple value of *max_seqlen_q*/*max_seqlen_k* by 128: $((x + 128 - 1) / 128 * 128)$.
12. Tensor *q*, *k*, *v*, *out*, *cu_seqlens_q*, *cu_seqlens_k*, *seqlen_k*, and *block_table* should have contiguous last dimension.
13. Data types of tensor *q*, *k*, *v*, *out*, *p* are only supported with CUDNN_DATA_HALF, CUDNN_DATA_FLOAT, CUDNN_DATA_BFLOAT16.
14. Data types of tensor *cu_seqlens_q*, *cu_seqlens_k*, *seqlen_k* and *block_table* must be CUDNN_DATA_INT32.

返回值

- **CUDNN_STATUS_SUCCESS**

The operation was launched successfully.

- **CUDNN_STATUS_BAD_PARAM**

At least one of the following conditions is met:

- At least one of the following params is NULL: *handle*, *q_desc*, *k_desc*, *v_desc*, *cu_seqlens_q_desc*, *cu_seqlens_k_desc*, *out_desc*, *q*, *k*, *v*, *cu_seqlens_q*, *cu_seqlens_k*, *out*.
- Data types of tensor *q*, *k*, *v*, *out* are not the same.
- When *return_softmax* is TRUE, *softmax_lse_desc*, *softmax_lse*, *p_desc*, *p* are NULL.
- *sequested_k_desc* & *sequested_k* are not both NULL or both not NULL.
- *block_table_desc* & *block_table_k* are not both NULL or both not NULL.
- *alibi_slopes_desc* & *alibi_slopes* are not both NULL or both not NULL.
- *softmax_lse_desc* & *softmax_lse* are not both NULL or both not NULL.
- *p_desc* & *p* are not both NULL or both not NULL.
- *workspace_size_in_bytes* is 0 while *workspace* is not NULL.
- Rules mentioned above in parameter descriptions are violated.
- **CUDNN_STATUS_EXECUTION_FAILED**

The function failed to launch on the GPU.

4.6.2 cudnnGetMHAVarlenForwardWorkspaceSize

```

cudnnStatus_t CUDNNWINAPI cudnnGetMHAVarlenForwardWorkspaceSize(
    cudnnHandle_t handle,
    const cudnnTensorDescriptor_t q_desc,
    const cudnnTensorDescriptor_t k_desc,
    const cudnnTensorDescriptor_t v_desc,
    const cudnnTensorDescriptor_t cu_seqlens_q_desc,
    const cudnnTensorDescriptor_t cu_seqlens_k_desc,
    const cudnnTensorDescriptor_t sequested_k_desc,
    const cudnnTensorDescriptor_t block_table_desc,
    const cudnnTensorDescriptor_t alibi_slopes_desc,
    const cudnnTensorDescriptor_t out_desc,
    const cudnnTensorDescriptor_t softmax_lse_desc,
    const cudnnTensorDescriptor_t p_desc,
    int max_seqlen_q,
    int max_seqlen_k,
    float dropout,
    float softmax_scale,
    bool zero_tensors,
    bool is_causal,
    int window_size_left,
    int window_size_right,
    bool return_softmax,
    size_t *size_in_bytes);

```

This function returns the amount of GPU memory workspace the user needs to allocate to be able to call `cudnnMHAVarlenForward()`.

参数

- **handle**

Input. Handle to a previously created cuDNN context. For more information, refer to `cudnnHandle_t`.

- **q_desc**

Input. Handle to a previously initialized query tensor descriptor. For more information, refer to `cudaTensorDescriptor_t`.

- **k_desc**

Input. Handle to a previously initialized key tensor descriptor. For more information, refer to `cudaTensorDescriptor_t`.

- **v_desc**

Input. Handle to a previously initialized value tensor descriptor. For more information, refer to `cudaTensorDescriptor_t`.

- **cu_seqlens_q_desc**

Input. Handle to a previously initialized `cu_seqlens_q` tensor descriptor. For more information, refer to `cudaTensorDescriptor_t`.

- **cu_seqlens_k_desc**

Input. Handle to a previously initialized `cu_seqlens_k` tensor descriptor. For more information, refer to `cudaTensorDescriptor_t`.

- **sequested_k_desc**

Input. Handle to a previously initialized `sequested_k` tensor descriptor. For more information, refer to `cudaTensorDescriptor_t`.

This argument is optional and should be set as NULL if not applied.

- **block_table_desc**

Input. Handle to a previously initialized `block_table` tensor descriptor. For more information, refer to `cudaTensorDescriptor_t`.

This argument is optional and should be set as NULL if not applied.

- **alibi_slopes_desc**

Input. Handle to a previously initialized alibi slopes tensor descriptor. For more information, refer to `cudaTensorDescriptor_t`.

If alibi slopes is not applied, this argument can be NULL.

- **out_desc**

Input. Handle to a previously initialized attention output tensor descriptor.

- **softmax_lse_desc**

Input. Handle to a previously initialized `softmax_lse` tensor descriptor.

If `return_softmax` is false, this argument can be NULL.

- **p_desc**

Input. Handle to a previously initialized `p` tensor descriptor. For more information, refer to `cudaTensorDescriptor_t`.

If `return_softmax` is false, this argument can be NULL.

- **max_seqlen_q**

Input. Maximum query sequence length in the batch.

- **max_seqlen_k**

Input. Maximum key sequence length in the batch.

- **dropout**

Input. Dropout probability; if greater than 0.0, dropout is applied.

- **softmax_scale**

Input. The scaling of QK^T before applying softmax. Default to $1 / \sqrt{\text{headdim}}$.

- **zero_tensors**

Input. Reserved, currently not used.

- **is_causal**

Input. Whether to apply causal attention mask.

- **window_size_left**

Input. Implements sliding window local attention with *window_size_right* together.

- **window_size_right**

Input. Implements sliding window local attention with *window_size_left* together.

- **return_softmax**

Input. Whether to return tensor *softmax_lse* and *p*. Return softmax is only supported when dropout is applied (greater than 0.0).

- **size_in_bytes**

Output. Amount of GPU memory needed as workspace to be able to execute `cudaMHAVarlenForward()`.

说明

1. Tensor *q* and *out* should be in dims of $[\text{total_q}, \text{nheads_q}, \text{headdim}]$, where total_q = total number of query tokens in the batch.
2. If *block_table* is not applied, tensor *k* and tensor *v* should be in dims of $[\text{total_k}, \text{nheads_k}, \text{headdim}]$, where total_k = total number of key tokens in the batch. Otherwise, they should be in dims of $[\text{num_blocks}, \text{page_block_size}, \text{nheads_k}, \text{headdim}]$.
3. Tensor *cu_seqlens_q* & *cu_seqlens_k* should be in dims of $[1, 1, \text{batch_size}+1]$ (cuDNN asks for at least 3 dims for tensor).
4. Tensor *seqlen_k* should be in dims of $[1, 1, \text{batch_size}]$.
5. Tensor *block_table* should be in dims of $[1, \text{batch_size}, \text{max_num_blocks_per_seq}]$.
6. Tensor *alibi_slopes* should be in dims of $[1, 1, \text{nheads_q}]$ or $[1, \text{batch_size}, \text{nheads_q}]$ and data type of CUDNN_DATA_FLOAT.
7. Tensor *softmax_lse* should be in dims of $[1, \text{nheads_q}, \text{total_q}]$ and data type of CUDNN_DATA_FLOAT.
8. Tensor *p* is the same as the tensor *S_dmask* described in [flash attention repo interface](#),

which is in dims of $[\text{batch_size}, \text{nheads_q}, \text{seqlen_q_rounded}, \text{seqlen_kv_rounded}]$.

$\text{seqlen_q_rounded}/\text{seqlen_kv_rounded}$ are the round multiple value of $\text{max_seqlen_q}/\text{max_seqlen_k}$ by 128: $((x + 128 - 1) / 128 * 128)$.

9. Tensor *q*, *k*, *v*, *out*, *cu_seqlens_q*, *cu_seqlens_k*, *seqlen_k*, and *block_table* should have contiguous last dimension.

10. Data types of tensor q , k , v , out , p are only supported with CUDNN_DATA_HALF, CUDNN_DATA_FLOAT, CUDNN_DATA_BFLOAT16.
11. Data types of tensor $cu_seqLens_q$, $cu_seqLens_k$, $seqused_k$ and $block_table$ must be CUDNN_DATA_INT32.

返回值

- **CUDNN_STATUS_SUCCESS**

The operation was launched successfully.

- **CUDNN_STATUS_BAD_PARAM**

At least one of the following conditions is met:

- At least one of the following params is NULL: $_handle$, q_desc , k_desc , v_desc , $cu_seqLens_q_desc$, $cu_seqLens_k_desc$, out_desc , $size_in_bytes$.
- Data types of tensor q , k , v , out are not the same.
- When $return_softmax$ is TRUE, $softmax_lse_desc$, p_desc are NULL.
- Rules mentioned above in parameter descriptions are violated.

- **CUDNN_STATUS_EXECUTION_FAILED**

The function failed to launch on the GPU.

4.7 cudnnMHAVarlenBackward [placeholder]

4.8 cudnnMHAForwardKVCache

4.8.1 cudnnMHAForwardKVCache

```
cudnnStatus_t CUDNNWINAPI cudnnMHAForwardKVCache(
    cudnnHandle_t handle,
    const cudnnTensorDescriptor_t q_desc,
    const void *q,
    const cudnnTensorDescriptor_t kcache_desc,
    const void *kcache,
    const cudnnTensorDescriptor_t vcache_desc,
    const void *vcache,
    const cudnnTensorDescriptor_t k_desc,
    const void *k,
    const cudnnTensorDescriptor_t v_desc,
    const void *v,
    const cudnnTensorDescriptor_t seqLens_k_desc,
    const void *seqLens_k,
    const cudnnTensorDescriptor_t rotary_cos_desc,
    const void *rotary_cos,
    const cudnnTensorDescriptor_t rotary_sin_desc,
    const void *rotary_sin,
    const cudnnTensorDescriptor_t cache_batch_idx_desc,
```



```

const void *cache_batch_idx,
const cudnnTensorDescriptor_t block_table_desc,
const void *block_table,
const cudnnTensorDescriptor_t alibi_slopes_desc,
void *alibi_slopes,
const cudnnTensorDescriptor_t out_desc,
void *out,
const cudnnTensorDescriptor_t softmax_lse_desc,
void *softmax_lse,
float softmax_scale,
bool is_causal,
int window_size_left,
int window_size_right,
bool is_rotary_interleaved,
int num_splits,
void *workspace,
size_t workspace_size_in_bytes);

```

This function supports flash attention with kvcache computation defined by repo [flash-attention](#).

参数

- **handle**
Input. Handle to a previously created cuDNN context. For more information, refer to `cudnnHandle_t`.
- **q_desc**
Input. Handle to a previously initialized query tensor descriptor. For more information, refer to `cudnnTensorDescriptor_t`.
- **q**
Input. Data pointer to GPU memory associated with the query tensor descriptor `q_desc`.
- **kcach_desc**
Input. Handle to a previously initialized kcache tensor descriptor. For more information, refer to `cudnnTensorDescriptor_t`.
- **kcach**
Input. Data pointer to GPU memory associated with the kcache tensor descriptor `kcach_desc`.
- **vcach_desc**
Input. Handle to a previously initialized vcache tensor descriptor. For more information, refer to `cudnnTensorDescriptor_t`.
- **vcach**
Input. Data pointer to GPU memory associated with the vcache tensor descriptor `vcach_desc`.
- **k_desc**
Input. Handle to a previously initialized key tensor descriptor. For more information, refer to `cudnnTensorDescriptor_t`.
This argument is optional and should be set as NULL if not applied.
- **k**

Input. Data pointer to GPU memory associated with the key tensor descriptor `k_desc`.

This argument is optional and should be set as NULL if not applied.

- **v_desc**

Input. Handle to a previously initialized value tensor descriptor. For more information, refer to `cudaTensorDescriptor_t`.

This argument is optional and should be set as NULL if not applied.

- **v**

Input. Data pointer to GPU memory associated with the value tensor descriptor `v_desc`.

This argument is optional and should be set as NULL if not applied.

- **seqLens_k_desc**

Input. Handle to a previously initialized `seqLens_k` tensor descriptor. For more information, refer to `cudaTensorDescriptor_t`.

This argument is optional and should be set as NULL if not applied.

- **seqLens_k**

Input. Data pointer to GPU memory associated with the `seqLens_k` tensor descriptor `seqLens_k_desc`.

The sequence lengths of the KV cache. This argument is optional and should be set as NULL if not applied.

- **rotary_cos_desc**

Input. Handle to a previously initialized `rotary_cos` tensor descriptor. For more information, refer to `cudaTensorDescriptor_t`.

This argument is optional and should be set as NULL if not applied.

- **rotary_cos**

Input. Data pointer to GPU memory associated with the `rotary_cos` tensor descriptor `rotary_cos_desc`.

If not None, apply rotary embedding to k and q. Only applicable if k and v are passed in. `rotary_dim` must be divisible by 16.

This argument is optional and should be set as NULL if not applied.

- **rotary_sin_desc**

Input. Handle to a previously initialized `rotary_sin` tensor descriptor. For more information, refer to `cudaTensorDescriptor_t`.

This argument is optional and should be set as NULL if not applied.

- **rotary_sin**

Input. Data pointer to GPU memory associated with the `rotary_sin` tensor descriptor `rotary_sin_desc`.

Similar to `rotary_cos`.

This argument is optional and should be set as NULL if not applied.

- **cache_batch_idx_desc**

Input. Handle to a previously initialized `cache_batch_idx` tensor descriptor. For more information, refer to `cudaTensorDescriptor_t`.

This argument is optional and should be set as NULL if not applied.

- **cache_batch_idx**

Input. Data pointer to GPU memory associated with the `cache_batch_idx` tensor descriptor `cache_batch_idx_desc`.

The indices used to index into the KV cache.

If NULL, we assume that the batch indices are `[0, 1, 2, ..., batch_size - 1]`.

If the indices are not distinct, and k and v are provided, the values updated in the cache might come from any of the duplicate indices.

This argument is optional and should be set as NULL if not applied.

- **block_table_desc**

Input. Handle to a previously initialized block_table tensor descriptor. For more information, refer to cudnnTensorDescriptor_t.

This argument is optional and should be set as NULL if not applied.

- **block_table**

Input. Data pointer to GPU memory associated with the block_table tensor descriptor block_table_desc.

This argument is optional and should be set as NULL if not applied.

- **alibi_slopes_desc**

Input. Handle to a previously initialized alibi slopes tensor descriptor. For more information, refer to cudnnTensorDescriptor_t.

If alibi slopes is not applied, this argument can be NULL.

- **alibi_slopes**

Input. Data pointer to GPU memory associated with the alibi slopes tensor descriptor alibi_slopes_desc.

A bias of $(-alibi_slope * |i + seqlen_k - seqlen_q - j|)$ is added to the attention score of query i and key j.

If alibi slopes is not applied, this argument can be NULL.

- **out_desc**

Input. Handle to a previously initialized attention output tensor descriptor.

- **out**

Output. Data pointer to GPU memory associated with the attention output tensor descriptor out_desc that carries the result.

- **softmax_lse_desc**

Input. Handle to a previously initialized softmax_lse tensor descriptor.

This argument is optional and should be set as NULL if not applied.

- **softmax_lse**

Output. Data pointer to GPU memory associated with the softmax_lse tensor descriptor alibi_slopes_desc.

The logsumexp of each row of the matrix $QK^T * scaling$ (e.g., log of the softmax normalization factor).

This argument is optional and should be set as NULL if not applied.

- **softmax_scale**

Input. The scaling of QK^T before applying softmax. Default to $1 / \sqrt{headdim}$.

- **is_causal**

Input. Whether to apply causal attention mask.

- **window_size_left**

Input. Implements sliding window local attention with *window_size_right* together.

- **window_size_right**

Input. Implements sliding window local attention with *window_size_left* together.

- **is_rotary_interleaved**

Input. Only applicable if rotary_cos and rotary_sin are passed in.

If True, rotary embedding will combine dimensions 0 & 1, 2 & 3, etc. If False, rotary embedding will combine dimensions 0 & rotary_dim / 2, 1 & rotary_dim / 2 + 1 (i.e. GPT-NeoX style).

- **num_splits**

Input. If > 1, split the key/value into this many chunks along the sequence.

If num_splits == 1, we don't split the key/value. If num_splits == 0, we use a heuristic to automatically determine the number of splits.

- **workspace**

Input. Data pointer to GPU memory to a workspace needed to be able to execute the specified algorithm.

If no workspace is needed for a particular algorithm, that pointer can be nil.

- **workspace_size_in_bytes**

Input. Specifies the size in bytes of the provided workSpace.

说明

1. Supports multi-query and grouped-query attention (MQA/GQA) by passing in KV with fewer heads than Q. Note that the number of heads in Q **must be divisible** by the number of heads in KV.
2. If *is_causal* != True, the causal mask is aligned to the bottom right corner of the attention matrix.
3. If window_size != (-1, -1), implements sliding window local attention. Query at position i will only attend to keys between [i + seqlen_k - seqlen_q - window_size[0], i + seqlen_k - seqlen_q + window_size[1]] inclusive.
4. Tensor q and out should be in dims of [batch_size, seqlen_q, nheads_q, headdim].
5. Tensor kcache, vcache should be in dims of [batch_size_cache, seqlen_cache, nheads_k, headdim] if there's no block_table, or [num_blocks, page_block_size, nheads_k, headdim] if there's a block_table (i.e. paged KV cache) page_block_size must be a multiple of 256.
6. Tensor k, v should be in dims of [batch_size, seqlen_new, nheads_k, headdim] if applied.
7. Tensor rotary_cos, rotary_sin should be in dims of [1, seqlen_ro, rotary_dim/2] if applied, rotary_dim must be divisible by 16.
8. Tensor seqLens_k and catch_batch_idx should be in dims of [1, 1, batch_size] if applied.
9. Tensor block_table should be in dims of [1, batch_size, max_num_blocks_per_seq].
10. Tensor *alibi_slopes* should be in dims of [1, 1, nheads_q] or [1, batch_size, nheads_q] and data type of CUDNN_DATA_FLOAT.
11. Tensor *softmax_lse* should be in dims of [batch_size, nheads_q, seqlen_q] and data type of CUDNN_DATA_FLOAT.
12. Tensor kcache, vcache, k, v should have contiguous last dimension.
13. Data types of tensor q, k, v, kcache, vcache, out are only supported with CUDNN_DATA_HALF, CUDNN_DATA_FLOAT, CUDNN_DATA_BFLOAT16.
14. Data types of tensor seqLens_k, catch_batch_idx and block_table must be CUDNN_DATA_INT32.

返回值

- **CUDNN_STATUS_SUCCESS**

The operation was launched successfully.

- **CUDNN_STATUS_BAD_PARAM**

At least one of the following conditions is met:

- At least one of the following params is NULL: *handle*, *q_desc*, *kcache_desc*, *vcache_desc*, *out_desc*, *q*, *kcache*, *vcache*, *out*.
- Data types of tensor *q*, *k*, *v*, *kcache*, *vcache*, *out* are not the same.
- *k_desc* & *k* are not both NULL or both not NULL.
- *v_desc* & *v* are not both NULL or both not NULL.
- *seqLens_k_desc* & *seqLens_k* are not both NULL or both not NULL.
- *rotary_cos_desc* & *rotary_cos* are not both NULL or both not NULL.
- *rotary_sin_desc* & *rotary_sin* are not both NULL or both not NULL.
- *cache_batch_idx_desc* & *cache_batch_idx* are not both NULL or both not NULL.
- *block_table_desc* & *block_table_k* are not both NULL or both not NULL.
- *alibi_slopes_desc* & *alibi_slopes* are not both NULL or both not NULL.
- *softmax_lse_desc* & *softmax_lse* are not both NULL or both not NULL.
- *workspace_size_in_bytes* is 0 while *workspace* is not NULL.
- Rules mentioned above in parameter descriptions are violated.

- **CUDNN_STATUS_EXECUTION_FAILED**

The function failed to launch on the GPU.

4.8.2 cudnnGetMHAFowardKVCacheWorkspaceSize

```
cudnnStatus_t CUDNNWINAPI cudnnGetMHAFowardKVCacheWorkspaceSize(
    cudnnHandle_t handle,
    const cudnnTensorDescriptor_t q_desc,
    const cudnnTensorDescriptor_t kcache_desc,
    const cudnnTensorDescriptor_t vcache_desc,
    const cudnnTensorDescriptor_t k_desc,
    const cudnnTensorDescriptor_t v_desc,
    const cudnnTensorDescriptor_t seqLens_k_desc,
    const cudnnTensorDescriptor_t rotary_cos_desc,
    const cudnnTensorDescriptor_t rotary_sin_desc,
    const cudnnTensorDescriptor_t cache_batch_idx_desc,
    const cudnnTensorDescriptor_t block_table_desc,
    const cudnnTensorDescriptor_t alibi_slopes_desc,
    const cudnnTensorDescriptor_t out_desc,
    const cudnnTensorDescriptor_t softmax_lse_desc,
    float softmax_scale,
    bool is_causal,
    int window_size_left,
    int window_size_right,
    bool is_rotary_interleaved,
    int num_splits,
    size_t *size_in_bytes);
```

This function returns the amount of GPU memory workspace the user needs to allocate to be able to call `cudnnMHAFowardKVCache()`.

参数

- **handle**

Input. Handle to a previously created cuDNN context. For more information, refer to `cudaDnnHandle_t`.

- **q_desc**

Input. Handle to a previously initialized query tensor descriptor. For more information, refer to `cudaDnnTensorDescriptor_t`.

- **kcache_desc**

Input. Handle to a previously initialized kcache tensor descriptor. For more information, refer to `cudaDnnTensorDescriptor_t`.

- **vcache_desc**

Input. Handle to a previously initialized vcache tensor descriptor. For more information, refer to `cudaDnnTensorDescriptor_t`.

- **k_desc**

Input. Handle to a previously initialized key tensor descriptor. For more information, refer to `cudaDnnTensorDescriptor_t`.

This argument is optional and should be set as NULL if not applied.

- **v_desc**

Input. Handle to a previously initialized value tensor descriptor. For more information, refer to `cudaDnnTensorDescriptor_t`.

This argument is optional and should be set as NULL if not applied.

- **seqLens_k_desc**

Input. Handle to a previously initialized seqLens_k tensor descriptor. For more information, refer to `cudaDnnTensorDescriptor_t`.

This argument is optional and should be set as NULL if not applied.

- **rotary_cos_desc**

Input. Handle to a previously initialized rotary_cos tensor descriptor. For more information, refer to `cudaDnnTensorDescriptor_t`.

This argument is optional and should be set as NULL if not applied.

- **rotary_sin_desc**

Input. Handle to a previously initialized rotary_sin tensor descriptor. For more information, refer to `cudaDnnTensorDescriptor_t`.

This argument is optional and should be set as NULL if not applied.

- **cache_batch_idx_desc**

Input. Handle to a previously initialized cache_batch_idx tensor descriptor. For more information, refer to `cudaDnnTensorDescriptor_t`.

This argument is optional and should be set as NULL if not applied.

- **block_table_desc**

Input. Handle to a previously initialized block_table tensor descriptor. For more information, refer to `cudaDnnTensorDescriptor_t`.

This argument is optional and should be set as NULL if not applied.

- **alibi_slopes_desc**

Input. Handle to a previously initialized alibi slopes tensor descriptor. For more information, refer to `cudaTensorDescriptor_t`.

If alibi slopes is not applied, this argument can be NULL.

- **out_desc**

Input. Handle to a previously initialized attention output tensor descriptor.

- **softmax_lse_desc**

Input. Handle to a previously initialized softmax_lse tensor descriptor.

This argument is optional and should be set as NULL if not applied.

- **softmax_scale**

Input. The scaling of QK^T before applying softmax. Default to $1 / \sqrt{\text{headdim}}$.

- **is_causal**

Input. Whether to apply causal attention mask.

- **window_size_left**

Input. Implements sliding window local attention with *window_size_right* together.

- **window_size_right**

Input. Implements sliding window local attention with *window_size_left* together.

- **is_rotary_interleaved**

Input. Only applicable if *rotary_cos* and *rotary_sin* are passed in.

If True, rotary embedding will combine dimensions 0 & 1, 2 & 3, etc. If False, rotary embedding will combine dimensions 0 & $\text{rotary_dim} / 2$, 1 & $\text{rotary_dim} / 2 + 1$ (i.e. GPT-NeoX style).

- **num_splits**

Input. If > 1 , split the key/value into this many chunks along the sequence.

If $\text{num_splits} == 1$, we don't split the key/value. If $\text{num_splits} == 0$, we use a heuristic to automatically determine the number of splits.

- **size_in_bytes**

Output. Amount of GPU memory needed as workspace to be able to execute `cudaMHAForwardKVCache()`.

说明

1. Tensor *q* and *out* should be in dims of $[\text{batch_size}, \text{seqlen_q}, \text{nheads_q}, \text{headdim}]$.
2. Tensor *kcache*, *vcache* should be in dims of $[\text{batch_size_cache}, \text{seqlen_cache}, \text{nheads_k}, \text{headdim}]$ if there's no *block_table*, or $[\text{num_blocks}, \text{page_block_size}, \text{nheads_k}, \text{headdim}]$ if there's a *block_table* (i.e. paged KV cache) *page_block_size* must be a multiple of 256.
3. Tensor *k*, *v* should be in dims of $[\text{batch_size}, \text{seqlen_new}, \text{nheads_k}, \text{headdim}]$ if applied.
4. Tensor *rotary_cos*, *rotary_sin* should be in dims of $[1, \text{seqlen_ro}, \text{rotary_dim}/2]$ if applied, *rotary_dim* must be divisible by 16.
5. Tensor *seqLens_k* and *catch_batch_idx* should be in dims of $[1, 1, \text{batch_size}]$ if applied.
6. Tensor *block_table* should be in dims of $[1, \text{batch_size}, \text{max_num_blocks_per_seq}]$.

7. Tensor *alibi_slopes* should be in dims of [1, 1, nheads_q] or [1, batch_size, nheads_q] and data type of CUDNN_DATA_FLOAT.
8. Tensor *softmax_lse* should be in dims of [batch_size, nheads_q, seqlen_q] and data type of CUDNN_DATA_FLOAT.
9. Tensor *kcache*, *vcache*, *k*, *v* should have contiguous last dimension.
10. Data types of tensor *q*, *k*, *v*, *kcache*, *vcache*, *out* are only supported with CUDNN_DATA_HALF, CUDNN_DATA_FLOAT, CUDNN_DATA_BFLOAT16.
11. Data types of tensor *seqLens_k*, *catch_batch_idx* and *block_table* must be CUDNN_DATA_INT32.

返回值

- **CUDNN_STATUS_SUCCESS**

The operation was launched successfully.

- **CUDNN_STATUS_BAD_PARAM**

At least one of the following conditions is met:

- At least one of the following params is NULL: *handle*, *q_desc*, *kcache_desc*, *vcache_desc*, *out_desc*, *size_in_bytes*.
- Data types of tensor *q*, *k*, *v*, *kcache*, *vcache*, *out* are not the same.
- Rules mentioned above in parameter descriptions are violated.

- **CUDNN_STATUS_EXECUTION_FAILED**

The function failed to launch on the GPU.

4.9 cudnnMaxPoolingIndicesBackward [placeholder]

4.10 cudnnMaxPoolingIndicesForward [placeholder]

4.11 cudnnFusedMoeForwardInference [placeholder]