曦云® 系列通用计算 GPU

mcBLAS API 参考

CSRD-23012-020-F3_V03　　2024-03-15

沐曦专有和三级保密信息　　本文档受 NDA 管控

声明

更新记录

版本	日期	更新说明
V03	2024-03-15	新增曦云®系列 GPU 产品信息
V02	2023-12-29	描述性修改及格式修正
V01	2023-10-16	正式版本首次发布

目录

1 简介

mcBLAS 库是基于 MetaX MXMACA® 运行时的基本线性代数子程序（Basic Linear Algebra Subprograms，BLAS）的实现。它允许用户访问 MetaX GPU 的计算资源。

要使用 mcBLAS API，应用程序必须在 GPU 内存空间中分配所需的矩阵和向量并使用数据填充，调用所需 mcBLAS 函数的序列，然后将结果从 GPU 内存空间上传回主机。mcBLAS API 还提供了用于从 GPU 写入和检索数据的辅助函数。

mcBLAS 提供了与传统 BLAS 功能相似的高性能健壮实现，适用于在 GPU 上运行。

mcBLAS 以 C++14 和 MXMACA 编写。它使用 MetaX MXMACA 运行时以在 GPU 设备上运行。

mcBLAS API 是一个使用沙漏模式（Hourglass Pattern）的精简 C99 API。它包含：

- [Helper]、[Level1]、[Level2]、[Level3] 和 [BLAS-Like] BLAS 函数，具有批处理（batched）版本和跨步批处理（strided-batched）版本。

1.1 安装 mcBLAS

$$MACA\begin{cases} library & \begin{cases} mcBLAS \\ \vdots \end{cases} \\ \vdots \\ \vdots \end{cases}$$

mcBLAS 库随 MXMACA 工具包一起发布。MXMACA 工具包的安装，参见《曦云系列通用计算 GPU 快速上手指南》。安装后，确保设置了环境变量 MACA_PATH。

```
export MACA_PATH=/opt/maca
```

这里的 /opt/maca 路径为 MXMACA 工具包的默认安装路径。如果安装时选择了其它路径，则将 MACA_PATH 设置为该路径。下面列出了与 mcBLAS API 相关的文件。

```
#header location:
${MACA_PATH}/include/mcblas

#lib location:
${MACA_PATH}/lib/libmcblas.so
```

在使用 mcBLAS 库进行编译之前，请确保环境变量 ISU_FASTMODEL 设置为 1。

```
export ISU_FASTMODEL=1
```

1.2 Hello mcBLAS

有关示例代码参考，请参见下面的两个示例。它们展示了使用有两种索引样式的 mcBLAS 库 API 和 C 语言编写应用程序（示例 1. “使用 C 和 mcBLAS 的应用程序：基于 1 的索引”和示例 2. “使用 C 和 mcBLAS 的应用程序：基于 0 的索引”）。

```
#CMakeLists.txt
cmake_minimum_required(VERSION 3.5.0)
set(PROJECT_NAME example)
project(${PROJECT_NAME} VERSION 0.1.0)
set(CMAKE_BUILD_TYPE "Debug")
set(CMAKE_CXX_STANDARD 14)
INCLUDE_DIRECTORIES(ENV{MACA_PATH}/include/mcr)
INCLUDE_DIRECTORIES(ENV{MACA_PATH}/include/mcblas)
INCLUDE_DIRECTORIES(ENV{MACA_PATH}/include)
LINK_DIRECTORIES(ENV{MACA_PATH}/lib)

file(GLOB EXAMPLE_SRCS1 "example1.cpp")
file(GLOB EXAMPLE_SRCS2 "example2.cpp")

add_executable("example1" ${EXAMPLE_SRCS1})
add_executable("example2" ${EXAMPLE_SRCS2})

target_link_libraries("example1" mcblas mcruntime)
target_link_libraries("example2" mcblas mcruntime)
```

```
//示例 1. 使用 C 和 mcBLAS 的应用程序：基于 1 的索引
//example1.cpp
//-----------------------------------------------------------
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <mc_runtime.h>
#include "mcblas.h"
#define M 6
#define N 5
#define IDX2F(i,j,ld) ((((j)-1)*(ld))+((i)-1))

static __inline__ void modify (mcblasHandle_t handle, float *m, int ldm,
↪int n, int p, int q, float alpha, float beta){
      mcblasSscal (handle, n-q+1, &alpha, &m[IDX2F(p,q,ldm)], ldm);
      mcblasSscal (handle, ldm-p+1, &beta, &m[IDX2F(p,q,ldm)], 1);
}

int main (void){
      int mcStat;
      mcblasStatus_t stat;
      mcblasHandle_t handle;
      int i, j;
      float* devPtrA;
      float* a = 0;
      a = (float *)malloc (M * N * sizeof (*a));
      if (!a) {
         printf ("host memory allocation failed");
         return EXIT_FAILURE;
      }
```

(下页继续)

(续上页)

```
      for (j = 1; j <= N; j++) {
         for (i = 1; i <= M; i++) {
            a[IDX2F(i,j,M)] = (float)((i-1) * N + j);
         }
      }
      mcStat = mcMalloc ((void**)&devPtrA, M*N*sizeof(*a));
      if (mcStat != 0) {
         printf ("device memory allocation failed");
         return EXIT_FAILURE;
      }
      stat = mcblasCreate(&handle);
      if (stat != MCBLAS_STATUS_SUCCESS) {
         printf ("mcBLAS initialization failed\n");
         return EXIT_FAILURE;
      }
      stat = mcblasSetMatrix (M, N, sizeof(*a), a, M, devPtrA, M);
      if (stat != MCBLAS_STATUS_SUCCESS) {
         printf ("data download failed");
         mcFree (devPtrA);
         mcblasDestroy(handle);
         return EXIT_FAILURE;
      }
      modify (handle, devPtrA, M, N, 2, 3, 16.0f, 12.0f);
      stat = mcblasGetMatrix (M, N, sizeof(*a), devPtrA, M, a, M);
      if (stat != MCBLAS_STATUS_SUCCESS) {
         printf ("data upload failed");
         mcFree (devPtrA);
         mcblasDestroy(handle);
         return EXIT_FAILURE;
      }
      mcFree (devPtrA);
      mcblasDestroy(handle);
      for (j = 1; j <= N; j++) {
         for (i = 1; i <= M; i++) {
            printf ("%7.0f", a[IDX2F(i,j,M)]);
         }
         printf ("\n");
      }
      free(a);
      return 0;
}
```

```
//示例 2. 使用 C 和 mcBLAS 的应用程序：基于 0 的索引
//example2.cpp
//-----------------------------------------------------------
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <mc_runtime.h>
#include "mcblas.h"
#define M 6
#define N 5
#define IDX2C(i,j,ld) (((j)*(ld))+(i))

static __inline__ void modify (mcblasHandle_t handle, float *m, int ldm,
↪int n, int p, int q, float alpha, float beta){
```

(下页继续)

(续上页)

```
        mcblasSscal (handle, n-q, &alpha, &m[IDX2C(p,q,ldm)], ldm);
        mcblasSscal (handle, ldm-p, &beta, &m[IDX2C(p,q,ldm)], 1);
}

int main (void){
        int mcStat;
        mcblasStatus_t stat;
        mcblasHandle_t handle;
        int i, j;
        float* devPtrA;
        float* a = 0;
        a = (float *)malloc (M * N * sizeof (*a));
        if (!a) {
           printf ("host memory allocation failed");
           return EXIT_FAILURE;
        }
        for (j = 0; j < N; j++) {
           for (i = 0; i < M; i++) {
              a[IDX2C(i,j,M)] = (float)(i * N + j + 1);
           }
        }
        mcStat = mcMalloc ((void**)&devPtrA, M*N*sizeof(*a));
        if (mcStat != 0) {
           printf ("device memory allocation failed");
           return EXIT_FAILURE;
        }
        stat = mcblasCreate(&handle);
        if (stat != MCBLAS_STATUS_SUCCESS) {
           printf ("mcBLAS initialization failed\n");
           return EXIT_FAILURE;
        }
        stat = mcblasSetMatrix (M, N, sizeof(*a), a, M, devPtrA, M);
        if (stat != MCBLAS_STATUS_SUCCESS) {
           printf ("data download failed");
           mcFree (devPtrA);
           mcblasDestroy(handle);
           return EXIT_FAILURE;
        }
        modify (handle, devPtrA, M, N, 1, 2, 16.0f, 12.0f);
        stat = mcblasGetMatrix (M, N, sizeof(*a), devPtrA, M, a, M);
        if (stat != MCBLAS_STATUS_SUCCESS) {
           printf ("data upload failed");
           mcFree (devPtrA);
           mcblasDestroy(handle);
           return EXIT_FAILURE;
        }
        mcFree (devPtrA);
        mcblasDestroy(handle);
        for (j = 0; j < N; j++) {
           for (i = 0; i < M; i++) {
              printf ("%7.0f", a[IDX2C(i,j,M)]);
           }
           printf ("\n");
        }
        free(a);
        return EXIT_SUCCESS;
```

(下页继续)

(续上页)

```
}
```

1.3 数据布局

为了最大程度地兼容现有的 Fortran 环境，mcBLAS 库使用列主（column-major）格式存储和基于 1 的索引。由于 C 和 C++ 使用行主（row-major）格式存储，因此用这些语言编写的应用程序不能对二维数组使用原生数组语义。相反，应定义宏或内联函数（inline function）以基于一维数组实现矩阵。对于以机械方式移植到 C 的 Fortran 代码，可以选择保留基于 1 的索引，以避免变换循环（loop）。在这种情况下，可以通过以下的宏来计算行“i”和列“j”中矩阵元素的数组索引。

```
#define IDX2F(i,j,ld) ((((j)-1)*(ld))+((i)-1))
```

在这里，ld 是指矩阵的前导维度（leading dimension），在列主存储的情况下，它是已分配矩阵（即使只使用其子矩阵）的行数。对于原生编写的 C 和 C++ 代码，最有可能选择基于 0 的索引，在这种情况下，可以通过以下的宏来计算行“i”和列“j”中矩阵元素的数组索引。

```
#define IDX2C(i,j,ld) (((j)*(ld))+(i))
```

2 使用 mcBLAS API

2.1 mcBLAS 数据类型

2.1.1 mcblasHandle_t

`mcblasHandle_t` 是指向不透明结构的指针类型，该结构包含 mcBLAS 库的上下文。mcBLAS 库上下文必须使用 `mcblasCreate()` 进行初始化，并且返回的句柄必须传入所有后续库函数调用。需要在结束时使用 `mcblasDestroy()` 销毁此上下文。

2.1.2 mcblas_int

```
typedef int32_t mcblas_int;
```

通过 mcBLAS，根据 MetaX 硬件以指定 int。

2.1.3 mcblas_stride

```
typedef int64_t mcblas_stride;
```

跨步批处理函数中矩阵或向量之间的步幅（stride）。

2.1.4 mcblas_half

表示 16 位浮点数的结构。

2.1.5 mcComplex

类，表示具有单精度实数和虚数部分的复数。

2.1.6 mcDoubleComplex

类，表示具有双精度实数和虚数部分的复数。

2.1.7 mcblasStatus_t

该类型用于函数状态返回。所有 mcBLAS 库函数调用返回其状态 mcblasStatus_t。

值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_ALLOC_FAILED	资源分配失败
MCBLAS_STATUS_INVALID_VALUE	使用无效的数值作为参数
MCBLAS_STATUS_ARCH_MISMATCH	当前函数功能在本设备上不支持
MCBLAS_STATUS_MAPPING_ERROR	访问 GPU 内存空间失败
MCBLAS_STATUS_EXECUTION_FAILED	GPU 程序无法执行
MCBLAS_STATUS_INTERNAL_ERROR	内部操作失败
MCBLAS_STATUS_NOT_SUPPORTED	不支持所需功能

2.1.8 macaDataType_t

macaDataType_t 是指定数据精度的枚举类型。当数据引用（data reference）不携带 type 类型时（例如 void *），使用此枚举。

例如，可在 mcblasSgemmEx 例程中使用。

值	含义
MACA_R_16F	数据类型为 16-bit 实数半精度浮点
MACA_C_16F	数据类型为 16-bit 复数半精度浮点
MACA_R_16BF	数据类型为 16-bit 实数 bfloat16 浮点
MACA_C_16BF	数据类型为 16-bit 复数 bfloat16 浮点
MACA_R_32F	数据类型为 32-bit 实数单精度浮点
MACA_C_32F	数据类型为 32-bit 复数单精度浮点
MACA_R_64F	数据类型为 64-bit 实数双精度浮点
MACA_C_64F	数据类型为 64-bit 复数双精度浮点
MACA_R_8I	数据类型为 8-bit 实数有符号整数
MACA_C_8I	数据类型为 8-bit 复数有符号整数
MACA_R_8U	数据类型为 8-bit 实数无符号整数
MACA_C_8U	数据类型为 8-bit 复数无符号整数
MACA_R_32I	数据类型为 32-bit 实数有符号整数
MACA_C_32I	数据类型为 32-bit 复数有符号整数

2.1.9 mcblasOperation_t

mcblasOperation_t 类型表明需要使用稠密矩阵（dense matrix）执行的操作。其值对应于 Fortran 字符 ‘N’ 或 ‘n’（非转置），‘T’ 或 ‘t’（转置），以及 ‘C’ 或 ‘c’（共轭转置），这些常用作传统 BLAS 实现的参数。

值	含义
MCBLAS_OP_N	选择非转置操作
MCBLAS_OP_T	选择转置操作
MCBLAS_OP_C	选择共轭转置操作

2.1.10 mcblasFillMode_t

该类型表明填充稠密矩阵的哪个部分（下或上），因此函数应使用该部分。其值对应于 Fortran 字符 ‘L’ 或 ‘l’（下），以及 ‘U’ 或 ‘u’（上），这些常用作传统 BLAS 实现的参数。

值	含义
MCBLAS_FILL_MODE_LOWER	填充矩阵的下半部分
MCBLAS_FILL_MODE_UPPER	填充矩阵的上半部分
MCBLAS_FILL_MODE_FULL	填充完整的矩阵

2.1.11 mcblasDiagType_t

该类型表明稠密矩阵的主对角线是否为单位元素，因此函数不应处理或修改主对角线。其值对应于 Fortran 字符 ‘N’ 或 ‘n’（非单位，non-unit），以及 ‘U’ 或 ‘u’（单位，unit），这些常用作传统 BLAS 实现的参数。

值	含义
MCBLAS_DIAG_NON_UNIT	矩阵对角线具有非单位元素
MCBLAS_DIAG_UNIT	矩阵对角线具有单位元素

2.1.12 mcblasSideMode_t

该类型表明稠密矩阵位于由特定函数解出的矩阵方程的左侧还是右侧。其值对应于 Fortran 字符 ‘L’ 或 ‘l’（左），以及 ‘R’ 或 ‘r’（右），这些常用作传统 BLAS 实现的参数。

值	含义
MCBLAS_SIDE_LEFT	矩阵位于方程的左侧
MCBLAS_SIDE_RIGHT	矩阵位于方程的右侧

2.1.13 mcblasPointerMode_t

mcblasPointerMode_t 类型表明是否在主机或设备上通过引用传递标量值。需要指出的是，如果函数调用中存在多个标量值，则它们都必须符合相同的单指针模式（single pointer mode）。可以分别使用 mcblasSetPointerMode() 和 mcblasGetPointerMode() 例程对指针模式进行设置和检索。

值	含义
MCBLAS_POINTER_MODE_HOST	在主机上通过引用传递标量
MCBLAS_POINTER_MODE_DEVICE	在设备上通过引用传递标量

2.1.14 mcblasAtomicsMode_t

该类型表明是否可以使用 mcBLAS 例程的采用了原子操作的备用实现。可以分别使用 mcblasSetAtomicsMode() 和 mcblasGetAtomicsMode() 例程对原子模式进行设置和查询。

值	含义
MCBLAS_ATOMICS_NOT_ALLOWED	不允许使用原子操作
MCBLAS_ATOMICS_ALLOWED	允许使用原子操作

2.1.15 mcblasGemmAlgo_t

mcblasGemmAlgo_t 是一个枚举类型，用于指定 GPU 上矩阵-矩阵乘的算法。mcBLAS 具有以下算法选项：

值	含义
MCBLAS_GEMM_DEFAULT	应用启发式算法选择 GEMM 算法。
MCBLAS_GEMM_ALGO0 - MCBLAS_GEMM_ALGO23	显式选择算法 [0,23]。
MCBLAS_GEMM_DEFAULT_TENSOR_OP	应用启发式算法选择 GEMM 算法，同时允许使用精度降低的 MCBLAS_COMPUTE_32F_FAST_16F 内核（向后兼容）。
MCBLAS_GEMM_ALGO0_TENSOR_OP - MCBLAS_GEMM_ALGO15_TENSOR_OP	显式选择一个 Tensor Core GEMM 算法 [0,15]。允许使用精度降低的 MCBLAS_COMPUTE_32F_FAST_16F 内核（向后兼容）。

2.1.16 mcblasMath_t

mcblasMath_t 枚举类型用于在 mcblasSetMathMode() 中选择计算精度模式，如下定义。

值	含义
MCBLAS_DEFAULT_MATH	默认且性能最高的模式，使用的计算和中间结果存储精度至少与请求的尾数和指数位数相同。将尽可能使用 Tensor Core。
MCBLAS_PEDANTIC_MATH	该模式对所有计算阶段使用规定的精度和标准化算法，主要用于数值稳健性研究、测试和调试。该模式的性能可能不如其他模式。
MCBLAS_TF32_TENSOR_OP_MATH	使用 TF32 Tensor Core 实现单精度例程加速。
MCBLAS_MATH_DISALLOW_REDUCED_PRECISION_REDUCTION	在输出类型精度小于计算类型精度的混合精度例程中，强制矩阵乘法期间的任意缩减使用累加器类型（即计算类型），而不是输出类型。这是一个标志，通过按位或（bitwise or）运算，可以与其他任意值一起设置。
MCBLAS_TENSOR_OP_MATH	允许库尽可能使用 Tensor Core 操作。对于单精度 GEMM 例程，mcBLAS 将使用 MCBLAS_COMPUTE_32F_FAST_16F 计算类型。

2.1.17 mcblasComputeType_t

mcblasComputeType_t 枚举类型用于在 mcblasGemmEx 和 mcblasLtMatmul 中（包括所有批处理变量和跨步批处理变量），选择计算精度模式，如下定义。

值	含义
MCBLAS_COMPUTE_16F	默认且性能最高的模式，16-bit 半精度浮点以及所有计算和中间结果存储精度至少为 16-bit 半精度。将尽可能使用 Tensor Core。
MCBLAS_COMPUTE_16F_PEDANTIC	该模式对所有计算阶段使用 16-bit 半精度浮点标准化算法，主要用于数值稳健性研究、测试和调试。该模式的性能可能不如其他模式，因为它禁用了 Tensor Core。
MCBLAS_COMPUTE_32F	默认的 32-bit 单精度浮点，使用至少 32-bit 的计算和中间结果存储精度。
MCBLAS_COMPUTE_32F_PEDANTIC	对所有计算阶段使用 32-bit 单精度浮点运算，并禁用算法优化，如高斯复杂度归约（3M）。

下页继续

表 2.1 – 续上页

值	含义
MCBLAS_COMPUTE_32F_FAST_16F	允许库对 32-bit 输入和输出矩阵使用具有自动下变频（down-conversion）功能的 Tensor Core 和 16-bit 半精度计算。
MCBLAS_COMPUTE_32F_FAST_16BF	允许库对 32-bit 输入和输出矩阵使用具有自动下变频换功能的 Tensor Core 和 bfloat16 计算。
MCBLAS_COMPUTE_32F_FAST_TF32	允许库对将 32-bit 输入和输出矩阵使用 Tensor Core 和 TF32 计算。
MCBLAS_COMPUTE_64F	默认的 64-bit 双精度浮点，使用至少 64-bit 的计算和中间结果存储精度。
MCBLAS_COMPUTE_64F_PEDANTIC	对所有计算阶段使用 64-bit 双精度浮点运算，并禁用算法优化，如高斯复杂度归约（3M）。
MCBLAS_COMPUTE_32I	默认的 32-bit 整数模式，使用至少 32-bit 的计算和中间结果存储精度。
MCBLAS_COMPUTE_32I_PEDANTIC	对所有计算阶段使用 32-bit 整数运算。

2.2 mcBLAS 辅助函数

2.2.1 mcblasCreate()

```
mcblasStatus_t
mcblasCreate(mcblasHandle_t *handle)
```

该函数用于初始化 mcBLAS 库并创建一个不透明结构的句柄，该结构存放 mcBLAS 库上下文。该函数在主机和设备上分配硬件资源，必须在调用其他任何 mcBLAS 库之前对其进行调用。mcBLAS 库上下文与当前 MXMACA 设备绑定。要在多个设备上使用库，需要为每个设备创建一个 mcBLAS 句柄。此外，对于给定设备，可以创建多个具有不同配置的 mcBLAS 句柄。由于通过 `mcblasCreate()` 分配一些内部资源和通过 `mcblasDestroy()` 释放这些资源将隐式调用 `mcblasDeviceSynchronize()`，因此建议尽量减少 `mcblasCreate()/mcblasDestroy()` 的调用次数。对于使用相同设备但来自不同线程的多线程应用，推荐使用的编程模型是为每个线程创建一个 mcBLAS 句柄，并在线程的整个生命周期中使用该 mcBLAS 句柄。

返回值	含义
MCBLAS_STATUS_SUCCESS	初始化成功
MCBLAS_STATUS_NOT_INITIALIZED	MXMACA 运行时初始化失败
MCBLAS_STATUS_ALLOC_FAILED	资源无法分配
MCBLAS_STATUS_INVALID_VALUE	handle == NULL

2.2.2 mcblasDestroy()

```
mcblasStatus_t
mcblasDestroy(mcblasHandle_t handle)
```

此函数释放 mcBLAS 库使用的硬件资源。此函数通常是使用 mcBLAS 库的特定句柄进行的最后一次调用。由于通过 `mcblasCreate()` 分配一些内部资源和通过 `mcblasDestroy()` 释放这些资源将隐式调用 `mcblasDeviceSynchronize()`，因此建议尽量减少 `mcblasCreate()/mcblasDestroy()` 的调用次数。

返回值	含义
MCBLAS_STATUS_SUCCESS	释放成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化

2.2.3 mcblasGetVersion()

```
mcblasStatus_t
mcblasGetVersion(mcblasHandle_t handle, int *version)
```

此函数返回 mcBLAS 库的版本号。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_INVALID_VALUE	为库版本号提供的存储未初始化（NULL）

2.2.4 mcblasGetProperty()

```
mcblasStatus_t
mcblasGetProperty(libraryPropertyType type, int *value)
```

该函数返回内存中请求属性的值（由 value 指向）。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_INVALID_VALUE	无效的类型值 • 无效的类型值 • value == NULL

2.2.5 mcblasGetStatusName()

```
const char* mcblasGetStatusName(mcblasStatus_t status)
```

该函数返回给定状态的字符串表示。

返回值	含义
NULL 结尾字符串	status 的字符串表示

2.2.6 mcblasGetStatusString()

```
const char* mcblasGetStatusString(mcblasStatus_t status)
```

该函数返回给定状态的描述字符串。

返回值	含义
NULL 结尾的字符串	status 的描述

2.2.7 mcblasSetStream()

```
mcblasStatus_t
mcblasSetStream(mcblasHandle_t handle, mcStream_t streamId)
```

此函数设置 mcBLAS 库流，该库流将用于后续所有的 mcBLAS 库函数调用。如果未设置 mcBLAS 库流，则所有内核都使用默认的 NULL 流。特别是，此例程可用于更改内核启动（kernel launch）之间的流，然后将 mcBLAS 库流重置回 NULL。此外，此函数无条件地将 mcBLAS 库工作空间重置回默认的工作空间池（请参见mcblasSetWorkspace()）。

返回值	含义
MCBLAS_STATUS_SUCCESS	流设置成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化

2.2.8 mcblasSetWorkspace()

```
mcblasStatus_t
mcblasSetWorkspace(mcblasHandle_t handle, void *workspace,
                   size_t workspaceSizeInBytes)
```

此函数将 mcBLAS 库工作空间设置为用户拥有的设备缓冲区，该缓冲区将用于后续所有的 mcBLAS 库函数调用（在当前设置的流上）。如果未设置 mcBLAS 库工作空间，则所有内核都使用在 mcBLAS 上下文创建期间分配的默认工作空间池。特别是，此例程可用于更改内核启动之间的工作空间。工作空间指针必须至少为 256 字节对齐，否则将返回 MCBLAS_STATUS_INVALID_VALUE 错误。mcblasSetStream() 函数无条件地将 mcBLAS 库工作空间重置回默认的工作空间池。workspaceSizeInBytes 太小可能会导致某些例程失败并返回 MCBLAS_STATUS_ALLOC_FAILED 错误，或导致较大的性能回归。工作空间需大于或等于 16KiB，以防止 MCBLAS_STATUS_ALLOC_FAILED 错误，而较大的工作空间可以为某些例程提供性能优势。建议用户提供的工作空间至少为 4MiB（以匹配 mcBLAS 的默认工作空间池）。

返回值	含义
MCBLAS_STATUS_SUCCESS	流设置成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_INVALID_VALUE	workspace 指针不满足至少 256 字节对齐

2.2.9 mcblasGetStream()

```
mcblasStatus_t
mcblasGetStream(mcblasHandle_t handle, mcStream_t *streamId)
```

此函数获取用于后续所有 mcBLAS 库函数调用的 mcBLAS 库流。如果未设置 mcBLAS 库流，则所有内核都使用默认的 NULL 流。

返回值	含义
MCBLAS_STATUS_SUCCESS	流返回成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_INVALID_VALUE	streamId == NULL

2.2.10 mcblasGetPointerMode()

```
mcblasStatus_t
mcblasGetPointerMode(mcblasHandle_t handle, mcblasPointerMode_t *mode)
```

此函数获取 mcBLAS 库使用的指针模式。有关详细信息，请参见mcblasPointerMode_t 。

返回值	含义
MCBLAS_STATUS_SUCCESS	指针模式获取成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_INVALID_VALUE	mode == NULL

2.2.11 mcblasSetPointerMode()

```
mcblasStatus_t
mcblasSetPointerMode(mcblasHandle_t handle, mcblasPointerMode_t mode)
```

此函数设置 mcBLAS 库使用的指针模式。default 是在主机上通过引用传递的值。有关详细信息，请参见mcblasPointerMode_t。

返回值	含义
MCBLAS_STATUS_SUCCESS	指针模式设置成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_INVALID_VALUE	mode 不是 MCBLAS_POINTER_MODE_HOST 或 MCBLAS_POINTER_MODE_DEVICE

2.2.12 mcblasSetVector()

```
mcblasStatus_t
mcblasSetVector(int n, int elemSize,
                const void *x, int incx, void *y, int incy)
```

此函数将主机内存空间中 x 向量的 n 个元素复制到 GPU 内存空间中的 y 向量。假定两个向量中的元素为 elemSize 字节。源向量 x 连续元素之间的存储间距由 incx 提供，目标向量 y 的由 incy 提供。

假定二维矩阵为列主格式，如果向量是矩阵的一部分，则等于 1 的向量增量会访问该矩阵的（部分）列。同样，使用与矩阵前导维度相等的增量会导致访问该矩阵的（部分）行。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_INVALID_VALUE	参数 incx, incy, elemSize<=0
MCBLAS_STATUS_MAPPING_ERROR	访问 GPU 内存时出错

2.2.13 mcblasGetVector()

```
mcblasStatus_t
mcblasGetVector(int n, int elemSize,
                const void *x, int incx, void *y, int incy)
```

此函数将 GPU 内存空间中 x 向量的 n 个元素复制到主机内存空间中的 y 向量。假定两个向量中的元素为 elemSize 字节。源向量连续元素之间的存储间距由 incx 提供，目标向量 y 的由 incy 提供。

假定二维矩阵为列主格式，如果向量是矩阵的一部分，则等于 1 的向量增量会访问该矩阵的（部分）列。同样，使用与矩阵前导维度相等的增量会导致访问该矩阵的（部分）行。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_INVALID_VALUE	参数 incx, incy, elemSize<=0
MCBLAS_STATUS_MAPPING_ERROR	访问 GPU 内存时出错

2.2.14 mcblasSetMatrix()

```
mcblasStatus_t
mcblasSetMatrix(int rows, int cols, int elemSize,
                const void *A, int lda, void *B, int ldb)
```

此函数将主机内存空间 A 矩阵中包含 rows x cols 个元素的分块复制到 GPU 内存空间中的 B 矩阵。假定每个元素存储都需要 elemSize 个字节，并且两个矩阵都以列主格式存储，源矩阵 A 和目标矩阵 B 的前导维度分别在 lda 和 ldb 中给出。前导维度表明已分配矩阵的行数，即使只使用其子矩阵。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_INVALID_VALUE	参数 rows, cols<0 或 elemSize, lda, ldb<=0
MCBLAS_STATUS_MAPPING_ERROR	访问 GPU 内存时出错

2.2.15 mcblasGetMatrix()

```
mcblasStatus_t
mcblasGetMatrix(int rows, int cols, int elemSize,
                const void *A, int lda, void *B, int ldb)
```

此函数将 GPU 内存空间 A 矩阵中包含 rows x cols 个元素的分块复制到主机内存空间中的 B 矩阵。假定每个元素存储都需要 elemSize 个字节，并且两个矩阵都以列主格式存储，源矩阵 A 和目标矩阵 B 的前导维度分别在 lda 和 ldb 中给出。前导维度表明已分配矩阵的行数，即使只使用其子矩阵。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_INVALID_VALUE	参数 rows, cols<0 或 elemSize, lda, ldb<=0
MCBLAS_STATUS_MAPPING_ERROR	访问 GPU 内存时出错

2.2.16 mcblasSetVectorAsync()

```
mcblasStatus_t
mcblasSetVectorAsync(int n, int elemSize, const void *hostPtr, int incx,
                     void *devicePtr, int incy, mcStream_t stream)
```

此函数与 mcblasSetVector() 具有相同的功能，但数据传输是使用给定的 MXMACA 流参数异步完成的（相对于主机）。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_INVALID_VALUE	参数 incx, incy, elemSize<=0
MCBLAS_STATUS_MAPPING_ERROR	访问 GPU 内存时出错

2.2.17 mcblasGetVectorAsync()

```
mcblasStatus_t
mcblasGetVectorAsync(int n, int elemSize, const void *devicePtr, int incx,
                     void *hostPtr, int incy, mcStream_t stream)
```

此函数与 mcblasGetVector() 具有相同的功能，但数据传输是使用给定的 MXMACA 流参数异步完成的（相对于主机）。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_INVALID_VALUE	参数 incx, incy, elemSize<=0
MCBLAS_STATUS_MAPPING_ERROR	访问 GPU 内存时出错

2.2.18 mcblasSetMatrixAsync()

```
mcblasStatus_t
mcblasSetMatrixAsync(int rows, int cols, int elemSize, const void *A,
                     int lda, void *B, int ldb, mcStream_t stream)
```

此函数与 mcblasSetMatrix() 具有相同的功能，但数据传输是使用给定的 MXMACA 流参数异步完成的（相对于主机）。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_INVALID_VALUE	参数 rows, cols<0 或 elemSize, lda, ldb<=0
MCBLAS_STATUS_MAPPING_ERROR	访问 GPU 内存时出错

2.2.19 mcblasGetMatrixAsync()

```
mcblasStatus_t
mcblasGetMatrixAsync(int rows, int cols, int elemSize, const void *A,
                     int lda, void *B, int ldb, mcStream_t stream)
```

此函数与 mcblasGetMatrix() 具有相同的功能，但数据传输是使用给定的 MXMACA 流参数异步完成的（相对于主机）。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_INVALID_VALUE	参数 rows, cols<0 或 elemSize, lda, ldb<=0
MCBLAS_STATUS_MAPPING_ERROR	访问 GPU 内存时出错

2.2.20 mcblasSetAtomicsMode()

```
mcblasStatus_t mcblasSetAtomicsMode(mcblasHandlet handle,
↪mcblasAtomicsMode_t mode)
```

有些例程，如 mcblas<t>symv 和 mcblas<t>hemv，具有备用实现（使用原子操作累加结果）。这种实现通常要快得多，但每次运行产生的结果可能不完全相同。从数学上讲，这些不同结果的差异并不显著，但在调试时，这些差异可能是有害的。

此函数可以允许或禁止 mcBLAS 库中所有具有备用实现的例程使用原子操作。如果在任何 mcBLAS 例程的文档中未明确说明，则意味着该例程没有使用原子操作的备用实现。当禁用原子模式时，若在同一硬件上使用相同的参数调用一个 mcBLAS 例程，该 mcBLAS 例程应在每次运行产生相同的结果。

默认初始化 mcblasHandle_t 对象的默认原子模式为 MCBLAS_ATOMICS_NOT_ALLOWED。有关详细信息，请参见mcBLAS 数据类型。

返回值	含义
MCBLAS_STATUS_SUCCESS	原子模式设置成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化

2.2.21 mcblasGetAtomicsMode()

```
mcblasStatus_t mcblasGetAtomicsMode(mcblasHandle_t handle,
↪mcblasAtomicsMode_t *mode)
```

此函数查询特定 mcBLAS 上下文的原子模式。

默认初始化 mcblasHandle_t 对象的默认原子模式为 MCBLAS_ATOMICS_NOT_ALLOWED。有关详细信息，请参见mcBLAS 数据类型。

返回值	含义
MCBLAS_STATUS_SUCCESS	原子模式查询成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_INVALID_VALUE	参数 mode 为空指针

2.2.22 mcblasSetMathMode()

```
mcblasStatus_t mcblasSetMathMode(mcblasHandle_t handle, mcblasMath_t mode)
```

mcblasSetMathMode 函数支持选择计算精度模式，如 mcblasMath_t 定义。允许用户将计算精度模式设置为自己的逻辑组合 (MCBLAS_TENSOR_OP_MATH 除外)。例如，mcblasSetMathMode(handle, MCBLAS_DEFAULT_MATH | MCBLAS_MATH_DISALLOW_REDUCED_PRECISION_REDUCTION)。请注意，默认的数学模式为 MCBLAS_DEFAULT_MATH。

返回值	含义
MCBLAS_STATUS_SUCCESS	数学模式设置成功
MCBLAS_STATUS_INVALID_VALUE	为 mode 指定了无效值
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化

2.2.23 mcblasGetMathMode()

```
mcblasStatus_t mcblasGetMathMode(mcblasHandle_t handle, mcblasMath_t *mode)
```

此函数返回库例程使用的数学模式。

返回值	含义
MCBLAS_STATUS_SUCCESS	数学类型返回成功
MCBLAS_STATUS_INVALID_VALUE	mode 的值为空
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化

2.2.24 mcblasSetSmCountTarget()

目前不支持此例程。

2.2.25 mcblasGetSmCountTarget()

目前不支持此例程。

2.2.26 mcblasLoggerConfigure()

```
mcblasStatus_t mcblasLoggerConfigure(
      int             logIsOn,
      int             logToStdOut,
      int             logToStdErr,
      const char*     logFileName)
```

此函数在运行时配置日志记录。除了以上这种类型的配置，还可以使用特殊的环境变量配置日志记录，libmcblas 会检查这些变量：

- MCBLAS_LOGINFO_DBG -设置 env. 变量为 1，表示打开日志记录（默认情况下日志记录处于关闭状态）。
- MCBLAS_LOGDEST_DBG -设置 env. 变量对如何记录进行编码。stdout，stderr 分别表示将日志消息输出到 stdout 和 stderr。在另一种情况下，它指定文件的 filename。

参数

- logIsOn：输入。完全打开/关闭日志记录。默认情况下为关闭，通过调用 `mcblasSetLoggerCallback` 以用户定义的回调函数来打开。
- logToStdOut：输入。打开/关闭标准输出 I/O 流的日志记录。默认情况下为关闭。
- logToStdErr：输入。打开/关闭标准错误 I/O 流的日志记录。默认情况下为关闭。
- logFileName：输入。打开/关闭文件系统中文件（由其名称指定）的日志记录。mcblasLoggerConfigure 拷贝 logFileName 的内容。如果不需要这种类型的日志记录，该指针应为空。

返回值

MCBLAS_STATUS_SUCCESS 成功。

2.2.27 mcblasGetLoggerCallback()

```
mcblasStatus_t mcblasGetLoggerCallback(
      mcblasLogCallback* userCallback)
```

此函数检索函数指针，指针指向通过 mcblasSetLoggerCallback 安装的用户自定义回调函数，否则指针为零。

参数

userCallback 输出。指针，指向用户定义的回调函数。

返回值

MCBLAS_STATUS_SUCCESS 成功。

2.2.28 mcblasSetLoggerCallback()

```
mcblasStatus_t mcblasSetLoggerCallback(
      mcblasLogCallback   userCallback)
```

此函数通过公共 mcBLAS C API 安装用户自定义回调函数。

参数

userCallback 输入。指针，指向用户定义的回调函数。

返回值

`MCBLAS_STATUS_SUCCESS` 成功。

2.3 mcBLAS Level-1 函数

本章中，我们将介绍执行基于标量和向量操作的 Level-1 基本线性代数子程序（BLAS1）函数。我们将使用 <type> 表示类型，<t> 表示相应的短类型（short type），以更简洁清晰地表示执行的函数。除非另有说明，否则 <type> 和 <t> 的含义如下：

<type>	<t>	含义
`float`	s 或 S	实数单精度
`double`	d 或 D	实数双精度
`mcComplex`	c 或 C	复数单精度
`mcDoubleComplex`	z 或 Z	复数双精度

当函数的参数和返回值不同时（有时发生在复数输入中），<t> 也可以具有以下含义：Sc，Cs，Dz，Zd。

缩写 **Re**(.) 和 **Im**(.) 分别代表一个数字的实数部分和虚数部分。由于实数不存在虚数部分，我们将虚数部分认为是零，通常可以把它从方程式中舍弃。此外，$\bar{\alpha}$ 表示 α 的复共轭（complex conjugate）。一般来说，在整个文档中，小写希腊符号 α 和 β 表示标量，小写粗体英文字母 **x** 和 **y** 表示向量，大写英文字母 A、B、C 表示矩阵。

2.3.1 mcblasI<t>amax()

```
mcblasStatus_t mcblasIsamax(mcblasHandle_t handle, int n,
                            const float *x, int incx, int *result)
mcblasStatus_t mcblasIdamax(mcblasHandle_t handle, int n,
                            const double *x, int incx, int *result)
mcblasStatus_t mcblasIcamax(mcblasHandle_t handle, int n,
                            const mcComplex *x, int incx, int *result)
mcblasStatus_t mcblasIzamax(mcblasHandle_t handle, int n,
                            const mcDoubleComplex *x, int incx, int
↪*result)
```

该函数查找最大量值元素的(最小)索引。因此，结果为第一个 i，使得在 $i = 1, \ldots, n$ 和 $j = 1+(i-1)*$ incx 时 |**Im** $(x[j])$|+|**Re** $(x[j])$| 最大。请注意，最后一个公式反映了基于 1 的索引，该索引用于与 Fortran 兼容。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
n		input	向量 x 中的元素数
x	设备	input	带元素的 <type> 向量
incx		input	x 中连续元素之间的步幅
result	主机或设备	output	结果索引，如果 `n,incx<=0` ，则为 0

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_ALLOC_FAILED	无法分配归约缓冲区
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.3.2 mcblasI<t>amin()

```
mcblasStatus_t mcblasIsamin(mcblasHandle_t handle, int n,
                            const float *x, int incx, int *result)
mcblasStatus_t mcblasIdamin(mcblasHandle_t handle, int n,
                            const double *x, int incx, int *result)
mcblasStatus_t mcblasIcamin(mcblasHandle_t handle, int n,
                            const mcComplex *x, int incx, int *result)
mcblasStatus_t mcblasIzamin(mcblasHandle_t handle, int n,
                            const mcDoubleComplex *x, int incx, int
↪*result)
```

该函数查找最小量值元素的(最小)索引。因此，结果为第一个 i，使得在 $i = 1, \ldots, n$ 和 $j = 1+(i-1)*$incx 时 $|\mathbf{Im}(x[j])|+|\mathbf{Re}(x[j])|$ 最小。请注意，最后一个公式反映了基于 1 的索引，该索引用于与 Fortran 兼容。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
n		input	向量 x 中的元素数
x	设备	input	带元素的 <type> 向量
incx		input	x 中连续元素之间的步幅
result	主机或设备	output	结果索引，如果 n,incx<=0 ，则为 0

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_ALLOC_FAILED	无法分配归约缓冲区
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.3.3 mcblas<t>asum()

```
mcblasStatus_t   mcblasSasum(mcblasHandle_t handle, int n,
                             const float           *x, int incx, float
↪*result)
mcblasStatus_t   mcblasDasum(mcblasHandle_t handle, int n,
                             const double          *x, int incx, double
↪*result)
mcblasStatus_t  mcblasScasum(mcblasHandle_t handle, int n,
                             const mcComplex       *x, int incx, float
↪*result)
mcblasStatus_t  mcblasDzasum(mcblasHandle_t handle, int n,
                             const mcDoubleComplex *x, int incx, double
↪*result)
```

该函数计算向量 x 中元素绝对值的总和。因此，结果为 $\sum_{i=1}^{n}|\mathbf{Im}(x[j])|+|\mathbf{Re}(x[j])|$，其中 $j = 1+(i-1)*$ incx。请注意，最后一个公式反映了基于 1 的索引，该索引用于与 Fortran 兼容。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
n		input	向量 x 中的元素数
x	设备	input	带元素的 <type> 向量
incx		input	x 中连续元素之间的步幅
result	主机或设备	output	结果索引，如果 n,incx<=0 ，则为 0.0

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_ALLOC_FAILED	无法分配归约缓冲区
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.3.4 mcblas<t>axpy()

```
mcblasStatus_t mcblasSaxpy(mcblasHandle_t handle, int n,
                           const float           *alpha,
                           const float           *x, int incx,
                           float                 *y, int incy)
mcblasStatus_t mcblasDaxpy(mcblasHandle_t handle, int n,
                           const double          *alpha,
                           const double          *x, int incx,
                           double                *y, int incy)
mcblasStatus_t mcblasCaxpy(mcblasHandle_t handle, int n,
                           const mcComplex       *alpha,
                           const mcComplex       *x, int incx,
                           mcComplex             *y, int incy)
mcblasStatus_t mcblasZaxpy(mcblasHandle_t handle, int n,
                           const mcDoubleComplex *alpha,
                           const mcDoubleComplex *x, int incx,
                           mcDoubleComplex       *y, int incy)
```

该函数将向量 x 乘以标量 α，并将其添加到 y 向量中，用结果覆盖最新向量。因此，执行的操作为 $\mathbf{y}[j] = \alpha \times \mathbf{x}[k] + \mathbf{y}[j]$，其中 $i = 1,\ldots,n$，$k = 1+(i-1)*$ incx，$j = 1+(i-1)*$ incy。请注意，最后两个公式反映了基于 1 的索引，该索引用于与 Fortran 兼容。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
alpha	主机或设备	input	用于乘法的 <type> 标量
n		input	向量 x 和 y 中的元素数
x	设备	input	包含 n 个元素的 <type> 向量
incx		input	x 中连续元素之间的步幅
y	设备	in/out	包含 n 个元素的 <type> 向量
incy		input	y 中连续元素之间的步幅

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.3.5 mcblas<t>copy()

```
mcblasStatus_t mcblasScopy(mcblasHandle_t handle, int n,
                           const float           *x, int incx,
                           float                 *y, int incy)
mcblasStatus_t mcblasDcopy(mcblasHandle_t handle, int n,
                           const double          *x, int incx,
                           double                *y, int incy)
mcblasStatus_t mcblasCcopy(mcblasHandle_t handle, int n,
                           const mcComplex       *x, int incx,
                           mcComplex             *y, int incy)
mcblasStatus_t mcblasZcopy(mcblasHandle_t handle, int n,
                           const mcDoubleComplex *x, int incx,
                           mcDoubleComplex       *y, int incy)
```

此函数将向量 x 复制到向量 y。因此，执行的操作为 $\mathbf{y}[j] = \mathbf{x}[k]$，其中 $i = 1, \ldots, n$，$k = 1 + (i-1) * \text{incx}$，$j = 1 + (i-1) * \text{incy}$。请注意，最后两个公式反映了基于 1 的索引，该索引用于与 Fortran 兼容。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
n		input	向量 x 和 y 中的元素数
x	设备	input	包含 n 个元素的 <type> 向量
incx		input	x 中连续元素之间的步幅
y	设备	output	包含 n 个元素的 <type> 向量
incy		input	y 中连续元素之间的步幅

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.3.6 mcblas<t>dot()

```
mcblasStatus_t mcblasSdot (mcblasHandle_t handle, int n,
                           const float           *x, int incx,
                           const float           *y, int incy,
                           float           *result)
mcblasStatus_t mcblasDdot (mcblasHandle_t handle, int n,
                           const double          *x, int incx,
                           const double          *y, int incy,
                           double          *result)
mcblasStatus_t mcblasCdotu(mcblasHandle_t handle, int n,
                           const mcComplex       *x, int incx,
                           const mcComplex       *y, int incy,
                           mcComplex       *result)
```

(下页继续)

(续上页)

```
mcblasStatus_t mcblasCdotc(mcblasHandle_t handle, int n,
                           const mcComplex       *x, int incx,
                           const mcComplex       *y, int incy,
                           mcComplex       *result)
mcblasStatus_t mcblasZdotu(mcblasHandle_t handle, int n,
                           const mcDoubleComplex *x, int incx,
                           const mcDoubleComplex *y, int incy,
                           mcDoubleComplex *result)
mcblasStatus_t mcblasZdotc(mcblasHandle_t handle, int n,
                           const mcDoubleComplex *x, int incx,
                           const mcDoubleComplex *y, int incy,
                           mcDoubleComplex       *result)
```

该函数计算向量 x 和 y 的点积（dot product）。因此，结果为 $\sum_{i=1}^{n}(\mathbf{x}[k]\times\mathbf{y}[j])$，其中 $k=1+(i-1)*$incx，$j=1+(i-1)*$ incy。请注意，在第一个方程中，如果函数名称以字符“c”结尾，则应使用 x 向量元素的共轭。最后两个公式反映了基于 1 的索引，该索引用于与 Fortran 兼容。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
n		input	向量 x 和 y 中的元素数
x	设备	input	包含 n 个元素的 <type> 向量
incx		input	x 中连续元素之间的步幅
y	设备	input	包含 n 个元素的 <type> 向量
incy		input	y 中连续元素之间的步幅
result	主机或设备	output	结果点积，如果 n<=0 ，则为 0.0

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_ALLOC_FAILED	无法分配归约缓冲区
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.3.7 mcblas<t>nrm2()

```
mcblasStatus_t  mcblasSnrm2(mcblasHandle_t handle, int n,
                            const float           *x, int incx, float
 ↪*result)
mcblasStatus_t  mcblasDnrm2(mcblasHandle_t handle, int n,
                            const double          *x, int incx, double
 ↪*result)
mcblasStatus_t mcblasScnrm2(mcblasHandle_t handle, int n,
                            const mcComplex       *x, int incx, float
 ↪*result)
mcblasStatus_t mcblasDznrm2(mcblasHandle_t handle, int n,
                            const mcDoubleComplex *x, int incx, double
 ↪*result)
```

该函数计算向量 x 的欧几里德范数（Euclidean norm）。代码中使用多相模型累积来避免中间结果下溢和溢出，其结果相当于精确运算中的 $\sqrt{\sum_{i=1}^{n}(\mathbf{x}[j]\times\mathbf{x}[j])}$，其中 $j=1+(i-1)*$ incx。请注意，最后一个公式反映了基于 1 的索引，该索引用于与 Fortran 兼容。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
n		input	向量 x 中的元素数
x	设备	input	包含 n 个元素的 <type> 向量
incx		input	x 中连续元素之间的步幅
result	主机或设备	output	结果范数，如果 n,incx<=0 ，则为 0.0

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_ALLOC_FAILED	无法分配归约缓冲区
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.3.8 mcblas<t>rot()

```
mcblasStatus_t  mcblasSrot(mcblasHandle_t handle, int n,
                           float          *x, int incx,
                           float          *y, int incy,
                           const float  *c, const float          *s)
mcblasStatus_t  mcblasDrot(mcblasHandle_t handle, int n,
                           double         *x, int incx,
                           double         *y, int incy,
                           const double *c, const double         *s)
mcblasStatus_t  mcblasCrot(mcblasHandle_t handle, int n,
                           mcComplex      *x, int incx,
                           mcComplex      *y, int incy,
                           const float  *c, const mcComplex      *s)
mcblasStatus_t mcblasCsrot(mcblasHandle_t handle, int n,
                           mcComplex      *x, int incx,
                           mcComplex      *y, int incy,
                           const float  *c, const float          *s)
mcblasStatus_t  mcblasZrot(mcblasHandle_t handle, int n,
                           mcDoubleComplex *x, int incx,
                           mcDoubleComplex *y, int incy,
                           const double *c, const mcDoubleComplex *s)
mcblasStatus_t mcblasZdrot(mcblasHandle_t handle, int n,
                           mcDoubleComplex *x, int incx,
                           mcDoubleComplex *y, int incy,
                           const double *c, const double         *s)
```

该函数将吉文斯旋转 (Givens rotation) 矩阵 (即，在 x，y 平面上逆时针旋转，旋转角度由 cos(alpha)=c，sin(alpha)=s 定义)：

$G = \begin{pmatrix} c & s \\ -s & c \end{pmatrix}$ 应用到向量 x 和 y 中。

因此，结果为 $\mathbf{x}[k] = c \times \mathbf{x}[k] + s \times \mathbf{y}[j]$ 和 $\mathbf{y}[j] = -s \times \mathbf{x}[k] + c \times \mathbf{y}[j]$，其中 $k = 1 + (i-1) * \text{incx}$，$j = 1 + (i-1) * \text{incy}$。请注意，最后两个公式反映了基于 1 的索引，该索引用于与 Fortran 兼容。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
n		input	向量 x 和 y 中的元素数
x	设备	in/out	包含 n 个元素的 <type> 向量
incx		input	x 中连续元素之间的步幅
y	设备	in/out	包含 n 个元素的 <type> 向量
incy		input	y 中连续元素之间的步幅
c	主机或设备	input	旋转矩阵的余弦元素
s	主机或设备	input	旋转矩阵的正弦元素

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.3.9 mcblas<t>rotg()

```
mcblasStatus_t mcblasSrotg(mcblasHandle_t handle,
                           float          *a, float          *b,
                           float  *c, float          *s)
mcblasStatus_t mcblasDrotg(mcblasHandle_t handle,
                           double          *a, double         *b,
                           double *c, double         *s)
mcblasStatus_t mcblasCrotg(mcblasHandle_t handle,
                           mcComplex       *a, mcComplex      *b,
                           float  *c, mcComplex       *s)
mcblasStatus_t mcblasZrotg(mcblasHandle_t handle,
                           mcDoubleComplex *a, mcDoubleComplex *b,
                           double *c, mcDoubleComplex *s)
```

该函数创建吉文斯旋转矩阵

$$G = \begin{pmatrix} c & s \\ -s & c \end{pmatrix}$$

用以将一个 2×1 向量 $(a, b)^T$ 的第二项归零。然后，对于实数，我们可以写为

$$\begin{pmatrix} c & s \\ -s & c \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} r \\ 0 \end{pmatrix}$$

其中 $c^2 + s^2 = 1$，$r = a^2 + b^2$ 。参数 a 和 b 将分别用 r 和 z 覆盖。z 的值使得 c 和 s 可以使用以下规则进行恢复：

$$(c, s) = \begin{cases} (\sqrt{1 - z^2}, z) & \text{if } |z| < 1 \\ (0.0, 1.0) & \text{if } |z| = 1 \\ (1/z, \sqrt{1 - z^2}) & \text{if } |z| > 1 \end{cases}$$

对于复数，我们可以写为

$$\begin{pmatrix} c & s \\ -\bar{s} & c \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} r \\ 0 \end{pmatrix}$$

其中 $c^2 + (\bar{s} \times s) = 1$，$r = \frac{a}{|a|} \times \| (a, b)^T \|_2$，当 $a \neq 0$ 且 $r = b$ 或 $a = 0$ 时，$\| (a, b)^T \|_2 = \sqrt{|a|^2 + |b|^2}$。最后，使用 r 覆盖 a 参数。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
a	主机或设备	in/out	用 r 覆盖的 <type> 标量
b	主机或设备	in/out	用 z 覆盖的 <type> 标量
c	主机或设备	output	旋转矩阵的余弦元素
s	主机或设备	output	旋转矩阵的正弦元素

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.3.10 mcblas<t>rotm()

```
mcblasStatus_t mcblasSrotm(mcblasHandle_t handle, int n, float  *x, int
↪incx,
                           float  *y, int incy, const float*  param)
mcblasStatus_t mcblasDrotm(mcblasHandle_t handle, int n, double *x, int
↪incx,
                           double *y, int incy, const double* param)
```

该函数将 modified Givens 变换

$$H = \begin{pmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{pmatrix}$$

应用到向量 x 和 y 中。

因此，结果为 $\mathbf{x}[k] = h_{11} \times \mathbf{x}[k] + h_{12} \times \mathbf{y}[j]$ 和 $\mathbf{y}[j] = h_{21} \times \mathbf{x}[k] + h_{22} \times \mathbf{y}[j]$，其中 $k = 1 + (i-1) * \text{incx}$，$j = 1 + (i-1) * \text{incy}$。请注意，最后两个公式反映了基于 1 的索引，该索引用于与 Fortran 兼容。矩阵 H 的 h_{11}、h_{12}、h_{21}、h_{22}、param[2]、param[3]、param[4] 中。flag=param[0] 为矩阵 H 的项定义以下预定义值。

flag=-1.0	flag= 0.0	flag= 1.0	flag=-2.0
$\begin{pmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{pmatrix}$	$\begin{pmatrix} 1.0 & h_{12} \\ h_{21} & 1.0 \end{pmatrix}$	$\begin{pmatrix} h_{11} & 1.0 \\ -1.0 & h_{22} \end{pmatrix}$	$\begin{pmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \end{pmatrix}$

请注意，flag 所隐含的值-1.0、0.0、1.0 不存储在参数中。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
n		input	向量 x 和 y 中的元素数
x	设备	in/out	包含 n 个元素的 <type> 向量
incx		input	x 中连续元素之间的步幅
y	设备	in/out	包含 n 个元素的 <type> 向量
incy		input	y 中连续元素之间的步幅
param	主机或设备	input	包含 5 个元素的 <type> 向量，其中 param[0] 和 param[1-4] 包含标志和矩阵 H

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.3.11 mcblas<t>rotmg()

```
mcblasStatus_t mcblasSrotmg(mcblasHandle_t handle, float  *d1, float  *d2,
                            float  *x1, const float  *y1, float  *param)
mcblasStatus_t mcblasDrotmg(mcblasHandle_t handle, double *d1, double *d2,
                            double *x1, const double *y1, double *param)
```

该函数构建 modified Givens 变换

$$H = \begin{pmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{pmatrix}$$

用以将一个 2×1 向量 $\left(\sqrt{d1} * x1, \sqrt{d2} * y1\right)^T$ 的第二项归零。`flag=param[0]` 为矩阵 H 的项定义以下预定义值。

flag=-1.0	flag= 0.0	flag= 1.0	flag=-2.0
$\begin{pmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{pmatrix}$	$\begin{pmatrix} 1.0 & h_{12} \\ h_{21} & 1.0 \end{pmatrix}$	$\begin{pmatrix} h_{11} & 1.0 \\ -1.0 & h_{22} \end{pmatrix}$	$\begin{pmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \end{pmatrix}$

请注意，flag 所隐含的值-1.0、0.0、1.0 不存储在参数中。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
d1	主机或设备	in/out	退出时覆盖的 <type> 标量
d2	主机或设备	in/out	退出时覆盖的 <type> 标量
x1	主机或设备	in/out	退出时覆盖的 <type> 标量
y1	主机或设备	input	<type> 标量
param	主机或设备	output	包含 5 个元素的 <type> 向量，其中 `param[0]` 和 `param[1-4]` 包含标志和矩阵 H

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.3.12 mcblas<t>scal()

```
mcblasStatus_t  mcblasSscal(mcblasHandle_t handle, int n,
                            const float           *alpha,
                            float           *x, int incx)
mcblasStatus_t  mcblasDscal(mcblasHandle_t handle, int n,
                            const double          *alpha,
                            double          *x, int incx)
mcblasStatus_t  mcblasCscal(mcblasHandle_t handle, int n,
```

(下页继续)

(续上页)

```
                            const mcComplex       *alpha,
                            mcComplex       *x, int incx)
mcblasStatus_t mcblasCsscal(mcblasHandle_t handle, int n,
                            const float           *alpha,
                            mcComplex       *x, int incx)
mcblasStatus_t  mcblasZscal(mcblasHandle_t handle, int n,
                            const mcDoubleComplex *alpha,
                            mcDoubleComplex *x, int incx)
mcblasStatus_t mcblasZdscal(mcblasHandle_t handle, int n,
                            const double          *alpha,
                            mcDoubleComplex *x, int incx)
```

该函数按标量 α 缩放向量 x，并用结果覆盖该向量。因此，执行的操作为 $\mathbf{x}[j] = \alpha \times \mathbf{x}[j]$，其中 $i = 1, \ldots, n$，$j = 1 + (i-1) *$ incx。请注意，最后两个公式反映了基于 1 的索引，该索引用于与 Fortran 兼容。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
alpha	主机或设备	input	用于乘法的 <type> 标量
n		input	向量 x 中的元素数
x	设备	in/out	包含 n 个元素的 <type> 向量
incx		input	x 中连续元素之间的步幅

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.3.13 mcblas<t>swap()

```
mcblasStatus_t mcblasSswap(mcblasHandle_t handle, int n, float
↪*x,
                           int incx, float           *y, int incy)
mcblasStatus_t mcblasDswap(mcblasHandle_t handle, int n, double
↪*x,
                           int incx, double          *y, int incy)
mcblasStatus_t mcblasCswap(mcblasHandle_t handle, int n, mcComplex
↪*x,
                           int incx, mcComplex       *y, int incy)
mcblasStatus_t mcblasZswap(mcblasHandle_t handle, int n, mcDoubleComplex
↪*x,
                           int incx, mcDoubleComplex *y, int incy)
```

该函数互换向量 x 和 y 的元素。因此，执行的操作为 $\mathbf{y}[j] \Leftrightarrow \mathbf{x}[k]$，其中 $i = 1, \ldots, n$，$k = 1 + (i-1) *$ incx，$j = 1 + (i-1) *$ incy。请注意，最后两个公式反映了基于 1 的索引，该索引用于与 Fortran 兼容。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
n		input	向量 x 和 y 中的元素数
x	设备	in/out	包含 n 个元素的 <type> 向量
incx		input	x 中连续元素之间的步幅
y	设备	in/out	包含 n 个元素的 <type> 向量
incy		input	y 中连续元素之间的步幅

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.4 mcBLAS Level-2 函数

本章中，我们将介绍执行矩阵-向量操作的 Level-2 基本线性代数子程序（BLAS2）函数。

2.4.1 mcblas<t>gbmv()

```
mcblasStatus_t mcblasSgbmv(mcblasHandle_t handle, mcblasOperation_t trans,
                           int m, int n, int kl, int ku,
                           const float           *alpha,
                           const float           *A, int lda,
                           const float           *x, int incx,
                           const float           *beta,
                           float           *y, int incy)
mcblasStatus_t mcblasDgbmv(mcblasHandle_t handle, mcblasOperation_t trans,
                           int m, int n, int kl, int ku,
                           const double          *alpha,
                           const double          *A, int lda,
                           const double          *x, int incx,
                           const double          *beta,
                           double          *y, int incy)
mcblasStatus_t mcblasCgbmv(mcblasHandle_t handle, mcblasOperation_t trans,
                           int m, int n, int kl, int ku,
                           const mcComplex       *alpha,
                           const mcComplex       *A, int lda,
                           const mcComplex       *x, int incx,
                           const mcComplex       *beta,
                           mcComplex       *y, int incy)
mcblasStatus_t mcblasZgbmv(mcblasHandle_t handle, mcblasOperation_t trans,
                           int m, int n, int kl, int ku,
                           const mcDoubleComplex *alpha,
                           const mcDoubleComplex *A, int lda,
                           const mcDoubleComplex *x, int incx,
                           const mcDoubleComplex *beta,
                           mcDoubleComplex *y, int incy)
```

该函数执行带状矩阵（banded matrix）-向量乘法

$$\mathbf{y} = \alpha\, \mathrm{op}(A)\mathbf{x} + \beta\mathbf{y}$$

其中 A 是具有次对角线 kl 和超对角线 ku 的带状矩阵，**x** 和 **y** 是向量，α 和 β 是标量。此外，对于矩阵 A

$$op(A) = \begin{cases} A & if\ \ transa == MCBLAS\_OP\_N \\ A^T & if\ \ transa == MCBLAS\_OP\_T \\ A^H & if\ \ transa == MCBLAS\_OP\_C \end{cases}$$

带状矩阵 A 按列存储，主对角线存储在行 $ku+1$ 中（从第一个位置开始），第一个超对角线存储在行 ku 中（从第二个位置开始），第一个次对角线存储在行 $ku+2$ 中（从第一个位置开始）等。因此通常，元素

$A(i,j)$ 存储在 A(ku+1+i-j,j)，其中 $j = 1, \ldots, n$，$i \in [\max(1, j-ku), \min(m, j+kl)]$。此外，数组 A 中的元素，在概念上与带状矩阵中的元素不对应（左上 $ku \times ku$ 和右下 $kl \times kl$ 三角），也不会被引用。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
trans		input	非转置或共轭转置运算 op(A)
m		input	矩阵 A 的行数
n		input	矩阵 A 的列数
kl		input	矩阵 A 的次对角线数
ku		input	矩阵 A 的超对角线数
alpha	主机或设备	input	用于乘法的 <type> 标量
A	设备	input	<type> 数组，维度为 lda x n，其中 lda>=kl+ku+1
lda		input	用于存储矩阵 A 的二维数组的前导维度
x	设备	input	如果 transa == MCBLAS_OP_N，则为包含 n 个元素的 <type> 向量；否则，包含 m 个元素
incx		input	x 中连续元素之间的步幅
beta	主机或设备	input	用于乘法的 <type> 标量，如果 beta == 0，则 y 不必为有效输入
y	设备	in/out	如果 transa == MCBLAS_OP_N，则为包含 m 个元素的 <type> 向量；否则，包含 n 个元素
incy		input	y 中连续元素之间的步幅

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_INVALID_VALUE	• m, n, kl, ku < 0 • lda < (kl+ku+1) • incx, incy == 0 • trans != MCBLAS_OP_N, MCBLAS_OP_T, MCBLAS_OP_C • alpha, beta == NULL
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.4.2 mcblas<t>gemv()

```
mcblasStatus_t mcblasSgemv(mcblasHandle_t handle, mcblasOperation_t trans,
                           int m, int n,
                           const float           *alpha,
                           const float           *A, int lda,
                           const float           *x, int incx,
                           const float           *beta,
                           float           *y, int incy)
mcblasStatus_t mcblasDgemv(mcblasHandle_t handle, mcblasOperation_t trans,
                           int m, int n,
                           const double          *alpha,
                           const double          *A, int lda,
                           const double          *x, int incx,
                           const double          *beta,
                           double          *y, int incy)
mcblasStatus_t mcblasCgemv(mcblasHandle_t handle, mcblasOperation_t trans,
                           int m, int n,
```

(下页继续)

(续上页)

```
                            const mcComplex       *alpha,
                            const mcComplex       *A, int lda,
                            const mcComplex       *x, int incx,
                            const mcComplex       *beta,
                            mcComplex       *y, int incy)
mcblasStatus_t mcblasZgemv(mcblasHandle_t handle, mcblasOperation_t trans,
                            int m, int n,
                            const mcDoubleComplex *alpha,
                            const mcDoubleComplex *A, int lda,
                            const mcDoubleComplex *x, int incx,
                            const mcDoubleComplex *beta,
                            mcDoubleComplex *y, int incy)
```

该函数执行矩阵-向量乘法

$$y = \alpha\, op(A)x + \beta\, y$$

其中 A 是以列主格式存储的 $m \times n$ 矩阵，**x** 和 **y** 是向量，α 和 β 是标量。此外，对于矩阵 A

$$op(A) = \begin{cases} A & if\ trans == MCBLAS\_OP\_N \\ A^T & if\ trans == MCBLAS\_OP\_T \\ A^H & if\ trans == MCBLAS\_OP\_C \end{cases}$$

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
trans		input	非转置或共轭转置运算 op(A)
m		input	矩阵 A 的行数
n		input	矩阵 A 的列数
alpha	主机或设备	input	用于乘法的 <type> 标量
A	设备	input	<type> 数组，维度为 lda x n，其中 lda >= max(1,m)。在输入之前，数组 A 的前导 m 乘 n 部分必须包含系数矩阵。退出时不变更
lda		input	用于存储矩阵 A 的二维数组的前导维度。lda 必须至少为 max(1,m)
x	设备	input	如果 transa == MCBLAS_OP_N，则为至少包含 (1+(n-1)*abs(incx)) 个元素的 <type> 向量；否则，至少包含 (1+(m-1)*abs(incx)) 个元素
incx		input	x 中连续元素之间的步幅
beta	主机或设备	input	用于乘法的 <type> 标量，如果 beta == 0，则 y 不必为有效输入
y	设备	in/out	如果 transa == MCBLAS_OP_N，则为至少包含 (1+(m-1)*abs(incy)) 个元素的 <type> 向量；否则，至少包含 (1+(n-1)*abs(incy)) 个元素
incy		input	y 中连续元素之间的步幅

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_INVALID_VALUE	参数 m,n<0 或 incx,incy=0
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.4.3 mcblas<t>ger()

```
mcblasStatus_t  mcblasSger(mcblasHandle_t handle, int m, int n,
                           const float           *alpha,
                           const float           *x, int incx,
                           const float           *y, int incy,
                           float           *A, int lda)
mcblasStatus_t  mcblasDger(mcblasHandle_t handle, int m, int n,
                           const double          *alpha,
                           const double          *x, int incx,
                           const double          *y, int incy,
                           double          *A, int lda)
mcblasStatus_t mcblasCgeru(mcblasHandle_t handle, int m, int n,
                           const mcComplex       *alpha,
                           const mcComplex       *x, int incx,
                           const mcComplex       *y, int incy,
                           mcComplex       *A, int lda)
mcblasStatus_t mcblasCgerc(mcblasHandle_t handle, int m, int n,
                           const mcComplex       *alpha,
                           const mcComplex       *x, int incx,
                           const mcComplex       *y, int incy,
                           mcComplex       *A, int lda)
mcblasStatus_t mcblasZgeru(mcblasHandle_t handle, int m, int n,
                           const mcDoubleComplex *alpha,
                           const mcDoubleComplex *x, int incx,
                           const mcDoubleComplex *y, int incy,
                           mcDoubleComplex *A, int lda)
mcblasStatus_t mcblasZgerc(mcblasHandle_t handle, int m, int n,
                           const mcDoubleComplex *alpha,
                           const mcDoubleComplex *x, int incx,
                           const mcDoubleComplex *y, int incy,
                           mcDoubleComplex *A, int lda)
```

此函数执行秩一校正（rank-1 update）

$$A = \begin{cases} \alpha \mathbf{x}\mathbf{y}^T + A & \text{if ger(), geru() is called} \\ \alpha \mathbf{x}\mathbf{y}^H + A & \text{if gerc() is called} \end{cases}$$

其中 A 是以列主格式存储的 $m \times n$ 矩阵，**x** 和 **y** 是向量，α 是标量。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
m		input	矩阵 A 的行数
n		input	矩阵 A 的列数
alpha	主机或设备	input	用于乘法的 <type> 标量
x	设备	input	包含 m 个元素的 <type> 向量
incx		input	x 中连续元素之间的步幅
y	设备	input	包含 n 个元素的 <type> 向量
incy		input	y 中连续元素之间的步幅
A	设备	in/out	<type> 数组，维度为 lda x n，其中 lda >= max(1,m)
lda		input	用于存储矩阵 A 的二维数组的前导维度

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_INVALID_VALUE	参数 m,n<0 或 incx,incy=0
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.4.4 mcblas<t>sbmv()

```
mcblasStatus_t mcblasSsbmv(mcblasHandle_t handle, mcblasFillMode_t uplo,
                           int n, int k, const float  *alpha,
                           const float  *A, int lda,
                           const float  *x, int incx,
                           const float  *beta, float *y, int incy)
mcblasStatus_t mcblasDsbmv(mcblasHandle_t handle, mcblasFillMode_t uplo,
                           int n, int k, const double *alpha,
                           const double *A, int lda,
                           const double *x, int incx,
                           const double *beta, double *y, int incy)
```

该函数执行对称带状矩阵-向量乘法

$$y = \alpha Ax + \beta y$$

其中 A 是具有 k 次对角线和超对角线的 $n \times n$ 对称带状矩阵，**x** 和 **y** 是向量，α 和 β 是标量。

如果 uplo == MCBLAS_FILL_MODE_LOWER，对称带状矩阵 A 按列存储，矩阵的主对角线存储在第 1 行，第一个次对角线存储在第 2 行（从第一个位置开始），第二个次对角线存储在第 3 行（从第一个位置开始）等。因此通常，元素 $A(i,j)$ 存储在 A(1+i-j,j)，其中 $j = 1,\ldots,n$，$i \in [j, \min(m, j+k)]$。此外，数组 A 中的元素，在概念上与带状矩阵中的元素不对应（右下 $k \times k$ 三角），也不会被引用。

如果 uplo == MCBLAS_FILL_MODE_UPPER，对称带状矩阵 A 按列存储，矩阵的主对角线存储在行 k+1，第一个超对角线在行 k 中（从第二个位置开始），第二个超对角线在行 k-1 中（从第三个位置开始）等。因此通常，元素 $A(i,j)$ 存储在 A(1+k+i-j,j)，其中 $j = 1,\ldots,n$，$i \in [\max(1, j-k), j]$。此外，数组 A 中的元素，在概念上与带状矩阵中的元素不对应（左上角 $k \times k$ 三角），也不会被引用。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
uplo		input	指示存储的是矩阵 A 的上半部分还是下半部分，其对称部分未被引用，且是从存储的元素推断出来的
n		input	矩阵 A 的行数和列数
k		input	矩阵 A 的次对角线数和超对角线数
alpha	主机或设备	input	用于乘法的 <type> 标量
A	设备	input	<type> 数组，维度为 lda x n，其中 lda >= k+1
lda		input	用于存储矩阵 A 的二维数组的前导维度
x	设备	input	包含 n 个元素的 <type> 向量
incx		input	x 中连续元素之间的步幅
beta	主机或设备	input	用于乘法的 <type> 标量，如果 beta == 0，则 y 不必为有效输入
y	设备	in/out	包含 n 个元素的 <type> 向量
incy		input	y 中连续元素之间的步幅

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_INVALID_VALUE	参数 n,k<0 或 incx,incy=0
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.4.5 mcblas<t>spmv()

```
mcblasStatus_t mcblasSspmv(mcblasHandle_t handle, mcblasFillMode_t uplo,
                           int n, const float  *alpha, const float  *AP,
                           const float  *x, int incx, const float  *beta,
                           float  *y, int incy)
mcblasStatus_t mcblasDspmv(mcblasHandle_t handle, mcblasFillMode_t uplo,
                           int n, const double *alpha, const double *AP,
                           const double *x, int incx, const double *beta,
                           double *y, int incy)
```

该函数执行对称压缩矩阵（packed matrix）-向量乘法

$$y = \alpha Ax + \beta y$$

其中 A 是以压缩格式（packed format）存储的 $n \times n$ 对称矩阵，**x** 和 **y** 是向量，α 和 β 是标量。

如果 uplo == MCBLAS_FILL_MODE_LOWER，对称矩阵 A 下三角部分中的元素逐列压缩在一起，元素之间没有间隙，以便将元素 $A(i,j)$ 存储在 AP[i+((2*n-j+1)*j)/2]，其中 $j = 1, \ldots, n$，$i \geq j$。因此，压缩格式只需要 $\frac{n(n+1)}{2}$ 个元素进行存储。

如果 uplo == MCBLAS_FILL_MODE_UPPER，对称矩阵 A 上三角部分中的元素逐列压缩在一起，元素之间没有间隙，以便将元素 $A(i,j)$ 存储在 AP[i+(j*(j+1))/2]，其中 $j = 1, \ldots, n$，$i \leq j$。因此，压缩格式只需要 $\frac{n(n+1)}{2}$ 个元素进行存储。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
uplo		input	指示存储的是矩阵 A 的上半部分还是下半部分，其对称部分未被引用，且是从存储的元素推断出来的
n		input	矩阵 A 的行数和列数
alpha	主机或设备	input	用于乘法的 <type> 标量
AP	设备	input	以压缩格式存储 A 的 <type> 数组
x	设备	input	包含 n 个元素的 <type> 向量
incx		input	x 中连续元素之间的步幅
beta	主机或设备	input	用于乘法的 <type> 标量，如果 beta == 0，则 y 不必为有效输入
y	设备	input	包含 n 个元素的 <type> 向量
incy		input	y 中连续元素之间的步幅

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_INVALID_VALUE	参数 n<0 或 incx,incy=0
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.4.6 mcblas<t>spr()

```
mcblasStatus_t mcblasSspr(mcblasHandle_t handle, mcblasFillMode_t uplo,
                          int n, const float  *alpha,
                          const float  *x, int incx, float  *AP)
mcblasStatus_t mcblasDspr(mcblasHandle_t handle, mcblasFillMode_t uplo,
                          int n, const double *alpha,
                          const double *x, int incx, double *AP)
```

此函数执行压缩对称秩一校正

$$A = \alpha \mathbf{x}\mathbf{x}^T + A$$

其中 A 是以压缩格式存储的 $n \times n$ 对称矩阵，**x** 是向量，α 是标量。如果 uplo == MCBLAS_FILL_MODE_LOWER，对称矩阵 A 下三角部分中的元素逐列压缩在一起，元素之间没有间隙，以便将元素 $A(i,j)$ 存储在 AP[i+((2*n-j+1)*j)/2]，其中 $j = 1,\ldots,n$，$i \geq j$。因此，压缩格式只需要 $\frac{n(n+1)}{2}$ 个元素进行存储。如果 uplo == MCBLAS_FILL_MODE_UPPER，对称矩阵 A 上三角部分中的元素逐列压缩在一起，元素之间没有间隙，以便将元素 $A(i,j)$ 存储在 AP[i+(j*(j+1))/2]，其中 $j = 1,\ldots,n$，$i \leq j$。因此，压缩格式只需要 $\frac{n(n+1)}{2}$ 个元素进行存储。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
uplo		input	指示存储的是矩阵 A 的上半部分还是下半部分，其对称部分未被引用，且是从存储的元素推断出来的
n		input	矩阵 A 的行数和列数
alpha	主机或设备	input	用于乘法的 <type> 标量
x	设备	input	包含 n 个元素的 <type> 向量
incx		input	x 中连续元素之间的步幅
AP	设备	in/out	以压缩格式存储 A 的 <type> 数组

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_INVALID_VALUE	参数 n<0 或 incx,incy=0
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.4.7 mcblas<t>spr2()

```
mcblasStatus_t mcblasSspr2(mcblasHandle_t handle, mcblasFillMode_t uplo,
                           int n, const float  *alpha,
                           const float  *x, int incx,
                           const float  *y, int incy, float  *AP)
mcblasStatus_t mcblasDspr2(mcblasHandle_t handle, mcblasFillMode_t uplo,
                           int n, const double *alpha,
                           const double *x, int incx,
                           const double *y, int incy, double *AP)
```

此函数执行压缩对称秩二更新

$$A = \alpha \left(\mathbf{x}\mathbf{y}^T + \mathbf{y}\mathbf{x}^T\right) + A$$

其中 A 是以压缩格式存储的 $n \times n$ 对称矩阵，**x** 是向量，α 是标量。如果 uplo == MCBLAS_FILL_MODE_LOWER，对称矩阵 A 下三角部分中的元素逐列压缩在一起，元素之间没有间

隙，以便将元素 $A(i,j)$ 存储在 AP[i+((2*n-j+1)*j)/2]，其中 $j=1,\ldots,n$，$i \geq j$。因此，压缩格式只需要 $\frac{n(n+1)}{2}$ 个元素进行存储。如果 uplo == MCBLAS_FILL_MODE_UPPER，对称矩阵 A 上三角部分中的元素逐列压缩在一起，元素之间没有间隙，以便将元素 $A(i,j)$ 存储在 AP[i+(j*(j+1))/2]，其中 $j=1,\ldots,n$，$i \leq j$。因此，压缩格式只需要 $\frac{n(n+1)}{2}$ 个元素进行存储。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
uplo		input	指示存储的是矩阵 A 的上半部分还是下半部分，其对称部分未被引用，且是从存储的元素推断出来的
n		input	矩阵 A 的行数和列数
alpha	主机或设备	input	用于乘法的 <type> 标量
x	设备	input	包含 n 个元素的 <type> 向量
incx		input	x 中连续元素之间的步幅
y	设备	input	包含 n 个元素的 <type> 向量
incy		input	y 中连续元素之间的步幅
AP	设备	in/out	以压缩格式存储 A 的 <type> 数组

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_INVALID_VALUE	参数 n<0 或 incx,incy=0
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.4.8 mcblas<t>symv()

```
mcblasStatus_t mcblasSsymv(mcblasHandle_t handle, mcblasFillMode_t uplo,
                           int n, const float           *alpha,
                           const float          *A, int lda,
                           const float          *x, int incx,
                           const float          *beta,
                           float          *y, int incy)
mcblasStatus_t mcblasDsymv(mcblasHandle_t handle, mcblasFillMode_t uplo,
                           int n, const double          *alpha,
                           const double          *A, int lda,
                           const double          *x, int incx,
                           const double          *beta,
                           double          *y, int incy)
mcblasStatus_t mcblasCsymv(mcblasHandle_t handle, mcblasFillMode_t uplo,
                           int n, const mcComplex       *alpha,
                           /* host or device pointer */
                           const mcComplex       *A, int lda,
                           const mcComplex       *x, int incx,
                           const mcComplex       *beta,
                           mcComplex       *y, int incy)
mcblasStatus_t mcblasZsymv(mcblasHandle_t handle, mcblasFillMode_t uplo,
                           int n, const mcDoubleComplex *alpha,
                           const mcDoubleComplex *A, int lda,
                           const mcDoubleComplex *x, int incx,
                           const mcDoubleComplex *beta,
                           mcDoubleComplex *y, int incy)
```

该函数执行对称矩阵-向量乘法。

$$y = \alpha Ax + \beta y$$

其中 A 是以 lower 或 upper 模式存储的 $n \times n$ 对称矩阵，**x** 和 **y** 是向量，α 和 β 是标量。此函数具有更快的备用实现（使用原子操作），可通过 `mcblasSetAtomicsMode()` 启用。

有关使用原子操作的更多详细信息，请参见mcblasSetAtomicsMode()。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
uplo		input	指示存储的是矩阵 `A` 的上半部分还是下半部分，其对称部分未被引用，且是从存储的元素推断出来的
n		input	矩阵 `A` 的行数和列数
alpha	主机或设备	input	用于乘法的 <type> 标量
A	设备	input	<type> 数组，维度为 `lda x n`，其中 `lda>=max(1,n)`
lda		input	用于存储矩阵 `A` 的二维数组的前导维度
x	设备	input	包含 `n` 个元素的 <type> 向量
incx		input	`x` 中连续元素之间的步幅
beta	主机或设备	input	用于乘法的 <type> 标量，如果 `beta == 0`，则 `y` 不必为有效输入
y	设备	in/out	包含 `n` 个元素的 <type> 向量
incy		input	`y` 中连续元素之间的步幅

下面列出了此函数可能的返回值及其含义。

返回值	含义
`MCBLAS_STATUS_SUCCESS`	操作成功
`MCBLAS_STATUS_NOT_INITIALIZED`	库未初始化
`MCBLAS_STATUS_INVALID_VALUE`	参数 `n<0` 或 `incx,incy=0`
`MCBLAS_STATUS_EXECUTION_FAILED`	此函数在 GPU 上启用失败

2.4.9 mcblas<t>syr()

```
mcblasStatus_t mcblasSsyr(mcblasHandle_t handle, mcblasFillMode_t uplo,
                          int n, const float             *alpha,
                          const float           *x, int incx,
                          float           *A, int lda)
mcblasStatus_t mcblasDsyr(mcblasHandle_t handle, mcblasFillMode_t uplo,
                          int n, const double            *alpha,
                          const double          *x, int incx,
                          double          *A, int lda)
mcblasStatus_t mcblasCsyr(mcblasHandle_t handle, mcblasFillMode_t uplo,
                          int n, const mcComplex         *alpha,
                          const mcComplex       *x, int incx,
                          mcComplex       *A, int lda)
mcblasStatus_t mcblasZsyr(mcblasHandle_t handle, mcblasFillMode_t uplo,
                          int n, const mcDoubleComplex   *alpha,
                          const mcDoubleComplex *x, int incx,
                          mcDoubleComplex *A, int lda)
```

此函数执行对称秩一校正

$$A = \alpha \mathbf{x}\mathbf{x}^T + A$$

其中 A 是以列主格式存储的 $n \times n$ 对称矩阵，**x** 是向量，α 是标量。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
uplo		input	指示存储的是矩阵 A 的上半部分还是下半部分，其对称部分未被引用，且是从存储的元素推断出来的
n		input	矩阵 A 的行数和列数
alpha	主机或设备	input	用于乘法的 <type> 标量
x	设备	input	包含 n 个元素的 <type> 向量
incx		input	x 中连续元素之间的步幅
A	设备	in/out	<type> 数组，维度为 lda x n，其中 lda>=max(1,n)
lda		input	用于存储矩阵 A 的二维数组的前导维度

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_INVALID_VALUE	参数 n<0 或 incx=0
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.4.10 mcblas<t>syr2()

```
mcblasStatus_t mcblasSsyr2(mcblasHandle_t handle, mcblasFillMode_t uplo,
                           int n,   const float          *alpha,
                           const float          *x, int incx,
                           const float          *y, int incy,
                           float          *A, int lda)
mcblasStatus_t mcblasDsyr2(mcblasHandle_t handle, mcblasFillMode_t uplo,
                           int n,   const double          *alpha,
                           const double          *x, int incx,
                           const double          *y, int incy,
                           double          *A, int lda)
mcblasStatus_t mcblasCsyr2(mcblasHandle_t handle, mcblasFillMode_t uplo,
                           int n,   const mcComplex       *alpha,
                           const mcComplex       *x, int incx,
                           const mcComplex       *y, int incy,
                           mcComplex       *A, int lda)
mcblasStatus_t mcblasZsyr2(mcblasHandle_t handle, mcblasFillMode_t uplo,
                           int n,   const mcDoubleComplex *alpha,
                           const mcDoubleComplex *x, int incx,
                           const mcDoubleComplex *y, int incy,
                           mcDoubleComplex *A, int lda)
```

此函数执行对称秩二校正

$$A = \alpha \left(\mathbf{x}\mathbf{y}^T + \mathbf{y}\mathbf{x}^T \right) + A$$

其中 A 是以列主格式存储的 $n \times n$ 对称矩阵，**x** 和 **y** 是向量，α 是标量。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
uplo		input	指示存储的是矩阵 A 的上半部分还是下半部分，其对称部分未被引用，且是从存储的元素推断出来的
n		input	矩阵 A 的行数和列数
alpha	主机或设备	input	用于乘法的 <type> 标量
x	设备	input	包含 n 个元素的 <type> 向量
incx		input	x 中连续元素之间的步幅
y	设备	input	包含 n 个元素的 <type> 向量
incy		input	y 中连续元素之间的步幅
A	设备	in/out	<type> 数组，维度为 lda x n，其中 lda>=max(1,n)
lda		input	用于存储矩阵 A 的二维数组的前导维度

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_INVALID_VALUE	参数 n<0 或 incx=0,incy=0
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.4.11 mcblas<t>tbmv()

```
mcblasStatus_t mcblasStbmv(mcblasHandle_t handle, mcblasFillMode_t uplo,
                           mcblasOperation_t trans, mcblasDiagType_t diag,
                           int n, int k, const float           *A, int lda,
                           float           *x, int incx)
mcblasStatus_t mcblasDtbmv(mcblasHandle_t handle, mcblasFillMode_t uplo,
                           mcblasOperation_t trans, mcblasDiagType_t diag,
                           int n, int k, const double          *A, int lda,
                           double          *x, int incx)
mcblasStatus_t mcblasCtbmv(mcblasHandle_t handle, mcblasFillMode_t uplo,
                           mcblasOperation_t trans, mcblasDiagType_t diag,
                           int n, int k, const mcComplex       *A, int lda,
                           mcComplex       *x, int incx)
mcblasStatus_t mcblasZtbmv(mcblasHandle_t handle, mcblasFillMode_t uplo,
                           mcblasOperation_t trans, mcblasDiagType_t diag,
                           int n, int k, const mcDoubleComplex *A, int lda,
                           mcDoubleComplex *x, int incx)
```

该函数执行三角带状矩阵-向量乘法

$$x = op(A)x$$

其中 A 是三角带状矩阵，**x** 是向量。此外，对于矩阵 A

$$op(A) = \begin{cases} A & if\ \ trans == MCBLAS\_OP\_N \\ A^T & if\ \ trans == MCBLAS\_OP\_T \\ A^H & if\ \ trans == MCBLAS\_OP\_C \end{cases}$$

如果 uplo == MCBLAS_FILL_MODE_LOWER，三角带状矩阵 A 按列存储，矩阵的主对角线存储在第 1 行，第一个次对角线在第 2 行中（从第一个位置开始），第二个次对角线在第 3 行中（从第一个位置开始）等。因此通常，元素 $A(i,j)$ 存储在 A(1+i-j,j)，其中 $j = 1,\ldots,n$，$i \in [j, \min(m, j+k)]$。此外，数组 A 中的元素，在概念上与带状矩阵中的元素不对应（右下 $k \times k$ 三角），也不会被引用。

如果 uplo == MCBLAS_FILL_MODE_UPPER，三角带状矩阵 A 按列存储，矩阵的主对角线存储在行 k+1，第一个超对角线在行 k 中（从第二个位置开始），第二个超对角线在行 k-1 中（从第三个位置开始）等。因此通常，元素 $A(i,j)$ 存储在 A(1+k+i-j,j)，其中 $j = 1,\ldots,n$，$i \in [\max(1, j-k, j)]$。此外，数组 A 中的元素，在概念上与带状矩阵中的元素不对应（左上角 $k \times k$ 三角），也不会被引用。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
uplo		input	指示存储的是矩阵 A 的上半部分还是下半部分，另外半部分未被引用，且是从存储的元素推断出来的
trans		input	非转置或共轭转置运算 op(A)
diag		input	指示矩阵 A 主对角线上的元素是否是统一的且不应被访问
n		input	矩阵 A 的行数和列数
k		input	矩阵的次对角线和超对角线
A	设备	input	<type> 数组，维度为 lda x n，其中 lda>=k+1
lda		input	用于存储矩阵 A 的二维数组的前导维度
x	设备	in/out	包含 n 个元素的 <type> 向量
incx		input	x 中连续元素之间的步幅

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_INVALID_VALUE	参数 n,k<0 或 incx=0
MCBLAS_STATUS_ALLOC_FAILED	内部暂存内存分配失败
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.4.12 mcblas<t>tbsv()

```
mcblasStatus_t mcblasStbsv(mcblasHandle_t handle, mcblasFillMode_t uplo,
                           mcblasOperation_t trans, mcblasDiagType_t diag,
                           int n, int k, const float           *A, int lda,
                           float           *x, int incx)
mcblasStatus_t mcblasDtbsv(mcblasHandle_t handle, mcblasFillMode_t uplo,
                           mcblasOperation_t trans, mcblasDiagType_t diag,
                           int n, int k, const double          *A, int lda,
                           double          *x, int incx)
mcblasStatus_t mcblasCtbsv(mcblasHandle_t handle, mcblasFillMode_t uplo,
                           mcblasOperation_t trans, mcblasDiagType_t diag,
                           int n, int k, const mcComplex       *A, int lda,
                           mcComplex       *x, int incx)
mcblasStatus_t mcblasZtbsv(mcblasHandle_t handle, mcblasFillMode_t uplo,
                           mcblasOperation_t trans, mcblasDiagType_t diag,
                           int n, int k, const mcDoubleComplex *A, int lda,
                           mcDoubleComplex *x, int incx)
```

该函数使用以下公式求解三角带状线性方程组

$$\text{op}(A)\mathbf{x} = \mathbf{b}$$

其中 A 是三角带状矩阵，**x** 和 **b** 是向量。此外，对于矩阵 A

$$op(A) = \begin{cases} A & if\ transa == MCBLAS\_OP\_N \\ A^T & if\ transa == MCBLAS\_OP\_T \\ A^H & if\ transa == MCBLAS\_OP\_C \end{cases}$$

结束时求得的解 **x** 覆盖右侧 **b**。此函数中不包括奇异性或接近奇异性的测试。

如果 uplo == MCBLAS_FILL_MODE_LOWER，三角带状矩阵 A 按列存储，矩阵的主对角线存储在第 1 行，第一个次对角线在第 2 行中（从第一个位置开始），第二个次对角线在第 3 行中（从第一个位置开始）等。因此通常，元素 $A(i,j)$ 存储在 A(1+i-j,j)，其中 $j = 1,\ldots,n$，$i \in [j, \min(m, j+k)]$。此外，数组 A 中的元素，在概念上与带状矩阵中的元素不对应（右下 $k \times k$ 三角），也不会被引用。如果 uplo == MCBLAS_FILL_MODE_UPPER，三角带状矩阵 A 按列存储，矩阵的主对角线存储在行 k+1，第一个超对角线在行 k 中（从第二个位置开始），第二个超对角线在行 k-1 中（从第三个位置开始）等。因此通常，元素 $A(i,j)$ 存储在 A(1+k+i-j,j)，其中 $j = 1,\ldots,n$，$i \in [\max(1, j-k), j)]$。此外，数组 A 中的元素，在概念上与带状矩阵中的元素不对应（左上角 $k \times k$ 三角），也不会被引用。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
uplo		input	指示存储的是矩阵 A 的上半部分还是下半部分，另外半部分未被引用，且是从存储的元素推断出来的
trans		input	非转置或共轭转置运算 op(A)
diag		input	指示矩阵 A 主对角线上的元素是否是统一的且不应被访问
n		input	矩阵 A 的行数和列数
k		input	矩阵 A 的次对角线和超对角线
A	设备	input	<type> 数组，维度为 lda x n，其中 lda>=k+1
lda		input	用于存储矩阵 A 的二维数组的前导维度
x	设备	in/out	包含 n 个元素的 <type> 向量
incx		input	x 中连续元素之间的步幅

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_INVALID_VALUE	参数 n,k<0 或 incx=0
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.4.13 mcblas<t>tpmv()

```
mcblasStatus_t mcblasStpmv(mcblasHandle_t handle, mcblasFillMode_t uplo,
                           mcblasOperation_t trans, mcblasDiagType_t diag,
                           int n, const float           *AP,
                           float           *x, int incx)
mcblasStatus_t mcblasDtpmv(mcblasHandle_t handle, mcblasFillMode_t uplo,
                           mcblasOperation_t trans, mcblasDiagType_t diag,
                           int n, const double          *AP,
                           double          *x, int incx)
mcblasStatus_t mcblasCtpmv(mcblasHandle_t handle, mcblasFillMode_t uplo,
                           mcblasOperation_t trans, mcblasDiagType_t diag,
                           int n, const mcComplex       *AP,
                           mcComplex       *x, int incx)
mcblasStatus_t mcblasZtpmv(mcblasHandle_t handle, mcblasFillMode_t uplo,
                           mcblasOperation_t trans, mcblasDiagType_t diag,
                           int n, const mcDoubleComplex *AP,
                           mcDoubleComplex *x, int incx)
```

该函数执行三角压缩矩阵-向量乘法

$$x = op(A)x$$

其中 A 是以压缩格式存储的三角矩阵，**x** 是向量。此外，对于矩阵 A

$$op(A) = \begin{cases} A & if\ trans == MCBLAS\_OP\_N \\ A^T & if\ trans == MCBLAS\_OP\_T \\ A^H & if\ trans == MCBLAS\_OP\_C \end{cases}$$

如果 `uplo == MCBLAS_FILL_MODE_LOWER`，三角矩阵 A 下三角部分中的元素逐列压缩在一起，元素之间没有间隙，以便将元素 $A(i,j)$ 存储在 `AP[i+((2*n-j+1)*j)/2]`，其中 $j = 1, \ldots, n$，$i \geq j$。因此，压缩格式只需要 $\frac{n(n+1)}{2}$ 个元素进行存储。

如果 `uplo == MCBLAS_FILL_MODE_UPPER`，三角矩阵 A 上三角部分中的元素逐列压缩在一起，元素之间没有间隙，以便将元素 $A(i,j)$ 存储在 `AP[i+(j*(j+1))/2]`，其中 $A(i,j)$，$i \leq j$。因此，压缩格式只需要 $\frac{n(n+1)}{2}$ 个元素进行存储。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
uplo		input	指示存储的是矩阵 A 的上半部分还是下半部分，另外半部分未被引用，且是从存储的元素推断出来的
trans		input	非转置或共轭转置运算 op(A)
diag		input	指示矩阵 A 主对角线上的元素是否是统一的且不应被访问
n		input	矩阵 A 的行数和列数
AP	设备	input	以压缩格式存储 A 的 <type> 数组
x	设备	in/out	包含 n 个元素的 <type> 向量
incx		input	x 中连续元素之间的步幅

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_INVALID_VALUE	• n < 0 • incx == 0 • uplo != MCBLAS_FILL_MODE_UPPER , MCBLAS_FILL_MODE_LOWER • trans != MCBLAS_OP_N, MCBLAS_OP_T, MCBLAS_OP_C • diag != MCBLAS_DIAG_UNIT, MCBLAS_DIAG_NON_UNIT
MCBLAS_STATUS_ALLOC_FAILED	内部暂存内存分配失败
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.4.14 mcblas<t>tpsv()

```
mcblasStatus_t mcblasStpsv(mcblasHandle_t handle, mcblasFillMode_t uplo,
                           mcblasOperation_t trans, mcblasDiagType_t diag,
                           int n, const float           *AP,
                           float           *x, int incx)
mcblasStatus_t mcblasDtpsv(mcblasHandle_t handle, mcblasFillMode_t uplo,
                           mcblasOperation_t trans, mcblasDiagType_t diag,
                           int n, const double          *AP,
                           double          *x, int incx)
mcblasStatus_t mcblasCtpsv(mcblasHandle_t handle, mcblasFillMode_t uplo,
                           mcblasOperation_t trans, mcblasDiagType_t diag,
```

(下页继续)

(续上页)

```
                            int n, const mcComplex       *AP,
                            mcComplex       *x, int incx)
mcblasStatus_t mcblasZtpsv(mcblasHandle_t handle, mcblasFillMode_t uplo,
                            mcblasOperation_t trans, mcblasDiagType_t diag,
                            int n, const mcDoubleComplex *AP,
                            mcDoubleComplex *x, int incx)
```

该函数使用以下公式求解压缩三角线性方程组

$$\text{op}(A)\mathbf{x} = \mathbf{b}$$

其中 A 是以压缩格式存储的三角矩阵，**x** 和 **b** 是向量。此外，对于矩阵 A

$$op(A) = \begin{cases} A & if\ trans == MCBLAS\_OP\_N \\ A^T & if\ trans == MCBLAS\_OP\_T \\ A^H & if\ trans == MCBLAS\_OP\_C \end{cases}$$

结束时 **x** 覆盖右侧 **b**。此函数中不包括奇异性或接近奇异性的测试。

如果 `uplo == MCBLAS_FILL_MODE_LOWER`，三角矩阵 A 下三角部分中的元素逐列压缩在一起，元素之间没有间隙，以便将元素 $A(i,j)$ 存储在 `AP[i+((2*n-j+1)*j)/2]`，其中 $j = 1,\ldots,n, i \geq j$。因此，压缩格式只需要 $\frac{n(n+1)}{2}$ 个元素进行存储。如果 `uplo == MCBLAS_FILL_MODE_UPPER`，三角矩阵 A 上三角部分中的元素逐列压缩在一起，元素之间没有间隙，以便将元素 $A(i,j)$ 存储在 `AP[i+(j*(j+1))/2]`，其中 $j = 1,\ldots,n$，$i \leq j$。因此，压缩格式只需要 $\frac{n(n+1)}{2}$ 个元素进行存储。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
uplo		input	指示存储的是矩阵 `A` 的上半部分还是下半部分，另外半部分未被引用，且是从存储的元素推断出来的
trans		input	非转置或共轭转置运算 op(`A`)
diag		input	指示矩阵 `A` 主对角线上的元素是否是统一的且不应被访问
n		input	矩阵 `A` 的行数和列数
AP	设备	input	以压缩格式存储 `A` 的 <type> 数组
x	设备	in/out	包含 `n` 个元素的 <type> 向量
incx		input	`x` 中连续元素之间的步幅

下面列出了此函数可能的返回值及其含义。

返回值	含义
`MCBLAS_STATUS_SUCCESS`	操作成功
`MCBLAS_STATUS_NOT_INITIALIZED`	库未初始化
`MCBLAS_STATUS_INVALID_VALUE`	• `n < 0` • `incx = 0` • `trans != MCBLAS_OP_N, MCBLAS_OP_C, MCBLAS_OP_T` • `uplo != MCBLAS_FILL_MODE_LOWER, MCBLAS_FILL_MODE_UPPER` • `diag != MCBLAS_DIAG_UNIT, MCBLAS_DIAG_NON_UNIT`
`MCBLAS_STATUS_EXECUTION_FAILED`	此函数在 GPU 上启用失败

2.4.15 mcblas<t>trmv()

```
mcblasStatus_t mcblasStrmv(mcblasHandle_t handle, mcblasFillMode_t uplo,
                           mcblasOperation_t trans, mcblasDiagType_t diag,
                           int n, const float           *A, int lda,
                           float           *x, int incx)
mcblasStatus_t mcblasDtrmv(mcblasHandle_t handle, mcblasFillMode_t uplo,
                           mcblasOperation_t trans, mcblasDiagType_t diag,
                           int n, const double          *A, int lda,
                           double          *x, int incx)
mcblasStatus_t mcblasCtrmv(mcblasHandle_t handle, mcblasFillMode_t uplo,
                           mcblasOperation_t trans, mcblasDiagType_t diag,
                           int n, const mcComplex       *A, int lda,
                           mcComplex       *x, int incx)
mcblasStatus_t mcblasZtrmv(mcblasHandle_t handle, mcblasFillMode_t uplo,
                           mcblasOperation_t trans, mcblasDiagType_t diag,
                           int n, const mcDoubleComplex *A, int lda,
                           mcDoubleComplex *x, int incx)
```

该函数执行三角矩阵-向量乘法

$$\mathbf{x} = \mathrm{op}(A)\mathbf{x}$$

其中 A 是以 lower 或 upper 模式存储的三角矩阵（无论是否有主对角线），**x** 是向量。此外，对于矩阵 A

$$op(A) = \begin{cases} A & if\ \ trans == MCBLAS\_OP\_N \\ A^T & if\ \ trans == MCBLAS\_OP\_T \\ A^H & if\ \ trans == MCBLAS\_OP\_C \end{cases}$$

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
uplo		input	指示存储的是矩阵 `A` 的上半部分还是下半部分，另外半部分未被引用，且是从存储的元素推断出来的
trans		input	非转置或共轭转置运算 op(`A`)
diag		input	指示矩阵 `A` 主对角线上的元素是否是统一的且不应被访问
n		input	矩阵 `A` 的行数和列数
A	设备	input	<type> 数组，维度为 `lda x n`，其中 `lda>=max(1,n)`
lda		input	用于存储矩阵 `A` 的二维数组的前导维度
x	设备	in/out	包含 `n` 个元素的 <type> 向量
incx		input	`x` 中连续元素之间的步幅

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_INVALID_VALUE	• n < 0 • incx = 0 • trans != MCBLAS_OP_N, MCBLAS_OP_C, MCBLAS_OP_T • uplo != MCBLAS_FILL_MODE_LOWER, MCBLAS_FILL_MODE_UPPER • diag != MCBLAS_DIAG_UNIT, MCBLAS_DIAG_NON_UNIT • lda < max(1, n)
MCBLAS_STATUS_ALLOC_FAILED	内部暂存内存分配失败
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.4.16 mcblas<t>trsv()

```
mcblasStatus_t mcblasStrsv(mcblasHandle_t handle, mcblasFillMode_t uplo,
                           mcblasOperation_t trans, mcblasDiagType_t diag,
                           int n, const float           *A, int lda,
                           float           *x, int incx)
mcblasStatus_t mcblasDtrsv(mcblasHandle_t handle, mcblasFillMode_t uplo,
                           mcblasOperation_t trans, mcblasDiagType_t diag,
                           int n, const double          *A, int lda,
                           double          *x, int incx)
mcblasStatus_t mcblasCtrsv(mcblasHandle_t handle, mcblasFillMode_t uplo,
                           mcblasOperation_t trans, mcblasDiagType_t diag,
                           int n, const mcComplex       *A, int lda,
                           mcComplex       *x, int incx)
mcblasStatus_t mcblasZtrsv(mcblasHandle_t handle, mcblasFillMode_t uplo,
                           mcblasOperation_t trans, mcblasDiagType_t diag,
                           int n, const mcDoubleComplex *A, int lda,
                           mcDoubleComplex *x, int incx)
```

该函数使用以下公式求解三角线性方程组

$$\text{op}(A)\mathbf{x} = \mathbf{b}$$

其中 A 是以 lower 或 upper 模式存储的三角矩阵（无论是否有主对角线），**x** 和 **b** 是向量。此外，对于矩阵 A

$$op(A) = \begin{cases} A & if\ trans == MCBLAS\_OP\_N \\ A^T & if\ trans == MCBLAS\_OP\_T \\ A^H & if\ trans == MCBLAS\_OP\_C \end{cases}$$

结束时 **x** 覆盖右侧 **b**。此函数中不包括奇异性或接近奇异性的测试。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
uplo		input	指示存储的是矩阵 A 的上半部分还是下半部分，另外半部分未被引用，且是从存储的元素推断出来的
trans		input	非转置或共轭转置运算 op(A)
diag		input	指示矩阵 A 主对角线上的元素是否是统一的且不应被访问
n		input	矩阵 A 的行数和列数
A	设备	input	<type> 数组，维度为 lda x n，其中 lda>=max(1,n)
lda		input	用于存储矩阵 A 的二维数组的前导维度
x	设备	in/out	包含 n 个元素的 <type> 向量
incx		input	x 中连续元素之间的步幅

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_INVALID_VALUE	• n < 0 • incx = 0 • trans != MCBLAS_OP_N, MCBLAS_OP_C, MCBLAS_OP_T • uplo != MCBLAS_FILL_MODE_LOWER, MCBLAS_FILL_MODE_UPPER • diag != MCBLAS_DIAG_UNIT, MCBLAS_DIAG_NON_UNIT • lda < max(1, n)
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.4.17 mcblas<t>hemv()

```
mcblasStatus_t mcblasChemv(mcblasHandle_t handle, mcblasFillMode_t uplo,
                           int n, const mcComplex       *alpha,
                           const mcComplex       *A, int lda,
                           const mcComplex       *x, int incx,
                           const mcComplex       *beta,
                           mcComplex       *y, int incy)
mcblasStatus_t mcblasZhemv(mcblasHandle_t handle, mcblasFillMode_t uplo,
                           int n, const mcDoubleComplex *alpha,
                           const mcDoubleComplex *A, int lda,
                           const mcDoubleComplex *x, int incx,
                           const mcDoubleComplex *beta,
                           mcDoubleComplex *y, int incy)
```

该函数执行厄米矩阵（Hermitian matrix）-向量乘法

$$y = \alpha A x + \beta y$$

其中 A 是以 lower 或 upper 模式存储的 $n \times n$ 厄米矩阵，**x** 和 **y** 是向量，α 和 β 是标量。此函数具有更快的备用实现（使用原子操作），可通过 mcblasSetAtomicsMode() 启用。

有关使用原子操作的详细信息，请参见mcblasSetAtomicsMode()。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
uplo		input	指示存储的是矩阵 A 的上半部分还是下半部分，另外 Hermitian 部分未被引用，且是从存储的元素推断出来的
n		input	矩阵 A 的行数和列数
alpha	主机或设备	input	用于乘法的 <type> 标量
A	设备	input	<type> 数组，维度为 lda x n，其中 lda>=max(1,n)。假定对角线元素的虚数部分为 0
lda		input	用于存储矩阵 A 的二维数组的前导维度
x	设备	input	包含 n 个元素的 <type> 向量
incx		input	x 中连续元素之间的步幅
beta	主机或设备	input	用于乘法的 <type> 标量，如果 beta == 0，则 y 不必为有效输入
y	设备	in/out	包含 n 个元素的 <type> 向量
incy		input	y 中连续元素之间的步幅

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_INVALID_VALUE	• n < 0 • incx = 0 或 incy = 0 • uplo != MCBLAS_FILL_MODE_LOWER, MCBLAS_FILL_MODE_UPPER • lda < n
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.4.18 mcblas<t>hbmv()

```
mcblasStatus_t mcblasChbmv(mcblasHandle_t handle, mcblasFillMode_t uplo,
                           int n, int k, const mcComplex       *alpha,
                           const mcComplex       *A, int lda,
                           const mcComplex       *x, int incx,
                           const mcComplex       *beta,
                           mcComplex       *y, int incy)
mcblasStatus_t mcblasZhbmv(mcblasHandle_t handle, mcblasFillMode_t uplo,
                           int n, int k, const mcDoubleComplex *alpha,
                           const mcDoubleComplex *A, int lda,
                           const mcDoubleComplex *x, int incx,
                           const mcDoubleComplex *beta,
                           mcDoubleComplex *y, int incy)
```

该函数执行厄米带状矩阵-向量乘法

$$y = \alpha Ax + \beta y$$

其中 A 是具有 k 次对角线和超对角线的 $n \times n$ 厄米带状矩阵，**x** 和 **y** 是向量，α 和 β 是标量。

如果 uplo == MCBLAS_FILL_MODE_LOWER，厄米带状矩阵 A 按列存储，矩阵的主对角线存储在第 1 行，第一个次对角线在第 2 行中（从第一个位置开始），第二个次对角线在第 3 行中（从第一个位置开始）等。因此通常，元素 $A(i,j)$ 存储在 A(1+i-j,j)，其中 $j = 1, \ldots, n$，$i \in [j, \min(m, j+k)]$。此外，数组 A 中的元素，在概念上与带状矩阵中的元素不对应（右下 $k \times k$ 三角），也不会被引用。

如果 uplo == MCBLAS_FILL_MODE_UPPER，厄米带状矩阵 A 按列存储，矩阵的主对角线存储在行 k+1，第一个超对角线在行 k 中（从第二个位置开始），第二个超对角线在行 k-1 中（从第三个位置开始）等。因此通常，元素 $A(i,j)$ 存储在 A(1+k+i-j,j)，其中 $j = 1,\ldots,n$，$i \in [\max(1, j-k), j]$。此外，数组 A 中的元素，在概念上与带状矩阵中的元素不对应（左上角 $k \times k$ 三角），也不会被引用。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
uplo		input	指示存储的是矩阵 A 的上半部分还是下半部分，另外 Hermitian 部分未被引用，且是从存储的元素推断出来的
n		input	矩阵 A 的行数和列数
k		input	矩阵 A 的次对角线和超对角线
alpha	主机或设备	input	用于乘法的 <type> 标量
A	设备	input	<type> 数组，维度为 lda x n，其中 lda>=k+1。假定对角线元素的虚数部分为 0
lda		input	用于存储矩阵 A 的二维数组的前导维度
x	设备	input	包含 n 个元素的 <type> 向量
incx		input	x 中连续元素之间的步幅
beta	主机或设备	input	用于乘法的 <type> 标量，如果 beta == 0，则 y 不必为有效输入
y	设备	in/out	包含 n 个元素的 <type> 向量
incy		input	y 中连续元素之间的步幅

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_INVALID_VALUE	• n < 0 或 k < 0 • incx = 0 或 incy = 0 • uplo != MCBLAS_FILL_MODE_LOWER, MCBLAS_FILL_MODE_UPPER • lda < (k + 1) • alpha = NULL 或 beta = NULL
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.4.19 mcblas<t>hpmv()

```
mcblasStatus_t mcblasChpmv(mcblasHandle_t handle, mcblasFillMode_t uplo,
                           int n, const mcComplex       *alpha,
                           const mcComplex       *AP,
                           const mcComplex       *x, int incx,
                           const mcComplex       *beta,
                           mcComplex       *y, int incy)
mcblasStatus_t mcblasZhpmv(mcblasHandle_t handle, mcblasFillMode_t uplo,
                           int n, const mcDoubleComplex *alpha,
                           const mcDoubleComplex *AP,
                           const mcDoubleComplex *x, int incx,
                           const mcDoubleComplex *beta,
                           mcDoubleComplex *y, int incy)
```

该函数执行厄米压缩矩阵-向量乘法

$$\mathbf{y} = \alpha A\mathbf{x} + \beta\mathbf{y}$$

其中 A 是以压缩格式存储的 $n \times n$ 厄米矩阵，**x** 和 **y** 是向量，α 和 β 是标量。如果 `uplo == MCBLAS_FILL_MODE_LOWER`，厄米矩阵 A 下三角部分中的元素逐列压缩在一起，元素之间没有间隙，以便将元素 $A(i,j)$ 存储在 `AP[i+((2*n-j+1)*j)/2]`，其中 $j = 1, \ldots, n$，$i \geq j$。因此，压缩格式只需要 $\frac{n(n+1)}{2}$ 个元素进行存储。如果 `uplo == MCBLAS_FILL_MODE_UPPER`，厄米矩阵 A 上三角部分中的元素逐列压缩在一起，元素之间没有间隙，以便将元素 $A(i,j)$ 存储在 `AP[i+(j*(j+1))/2]`，其中 $j = 1, \ldots, n$，$i \leq j$。因此，压缩格式只需要 $\frac{n(n+1)}{2}$ 个元素进行存储。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
uplo		input	指示存储的是矩阵 `A` 的上半部分还是下半部分，另外 Hermitian 部分未被引用，且是从存储的元素推断出来的
n		input	矩阵 `A` 的行数和列数
alpha	主机或设备	input	用于乘法的 <type> 标量
AP	设备	input	以压缩格式存储 `A` 的 <type> 数组。假定对角线元素的虚数部分为 0
x	设备	input	包含 `n` 个元素的 <type> 向量
incx		input	`x` 中连续元素之间的步幅
beta	主机或设备	input	用于乘法的 <type> 标量，如果 `beta == 0`，则 `y` 不必为有效输入
y	设备	in/out	包含 `n` 个元素的 <type> 向量
incy		input	`y` 中连续元素之间的步幅

下面列出了此函数可能的返回值及其含义。

返回值	含义
`MCBLAS_STATUS_SUCCESS`	操作成功
`MCBLAS_STATUS_NOT_INITIALIZED`	库未初始化
`MCBLAS_STATUS_INVALID_VALUE`	• `n < 0` • `incx = 0` 或 `incy = 0` • `uplo != MCBLAS_FILL_MODE_UPPER, MCBLAS_FILL_MODE_LOWER` • `alpha = NULL` 或 `beta = NULL`
`MCBLAS_STATUS_EXECUTION_FAILED`	此函数在 GPU 上启用失败

2.4.20 mcblas<t>her()

```
mcblasStatus_t mcblasCher(mcblasHandle_t handle, mcblasFillMode_t uplo,
                          int n, const float  *alpha,
                          const mcComplex       *x, int incx,
                          mcComplex       *A, int lda)
mcblasStatus_t mcblasZher(mcblasHandle_t handle, mcblasFillMode_t uplo,
                          int n, const double *alpha,
                          const mcDoubleComplex *x, int incx,
                          mcDoubleComplex *A, int lda)
```

此函数执行厄米秩一校正

$$A = \alpha \mathbf{x}\mathbf{x}^H + A$$

其中 A 是以列主格式存储的 $n \times n$ 厄米矩阵，**x** 是向量，α 是标量。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
uplo		input	指示存储的是矩阵 A 的上半部分还是下半部分，另外 Hermitian 部分未被引用，且是从存储的元素推断出来的
n		input	矩阵 A 的行数和列数
alpha	主机或设备	input	用于乘法的 <type> 标量
x	设备	input	包含 n 个元素的 <type> 向量
incx		input	x 中连续元素之间的步幅
A	设备	in/out	<type> 数组，维度为 lda x n，其中 lda>=max(1,n)。假定对角线元素的虚数部分为 0
lda		input	用于存储矩阵 A 的二维数组的前导维度

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_INVALID_VALUE	• n < 0 • incx = 0 • uplo != MCBLAS_FILL_MODE_UPPER, MCBLAS_FILL_MODE_LOWER • lda < max(1, n) • alpha = NULL
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.4.21 mcblas<t>her2()

```
mcblasStatus_t mcblasCher2(mcblasHandle_t handle, mcblasFillMode_t uplo,
                           int n, const mcComplex       *alpha,
                           const mcComplex       *x, int incx,
                           const mcComplex       *y, int incy,
                           mcComplex       *A, int lda)
mcblasStatus_t mcblasZher2(mcblasHandle_t handle, mcblasFillMode_t uplo,
                           int n, const mcDoubleComplex *alpha,
                           const mcDoubleComplex *x, int incx,
                           const mcDoubleComplex *y, int incy,
                           mcDoubleComplex *A, int lda)
```

此函数执行厄米秩二校正

$$A = \alpha \mathbf{x}\mathbf{y}^H + \overline{\alpha}\mathbf{y}\mathbf{x}^H + A$$

其中 A 是以列主格式存储的 $n \times n$ 厄米矩阵，**x** 和 **y** 是向量，α 是标量。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
uplo		input	指示存储的是矩阵 A 的上半部分还是下半部分，另外 Hermitian 部分未被引用，且是从存储的元素推断出来的
n		input	矩阵 A 的行数和列数
alpha	主机或设备	input	用于乘法的 <type> 标量
x	设备	input	包含 n 个元素的 <type> 向量
incx		input	x 中连续元素之间的步幅
y	设备	input	包含 n 个元素的 <type> 向量
incy		input	y 中连续元素之间的步幅
A	设备	in/out	<type> 数组，维度为 lda x n，其中 lda>=max(1,n)。假定对角线元素的虚数部分为 0
lda		input	用于存储矩阵 A 的二维数组的前导维度

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_INVALID_VALUE	• n < 0 • incx = 0 或 incy = 0 • uplo != MCBLAS_FILL_MODE_UPPER, MCBLAS_FILL_MODE_LOWER • lda < max(1, n) • alpha = NULL
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.4.22 mcblas<t>hpr()

```
mcblasStatus_t mcblasChpr(mcblasHandle_t handle, mcblasFillMode_t uplo,
                          int n, const float *alpha,
                          const mcComplex       *x, int incx,
                          mcComplex       *AP)
mcblasStatus_t mcblasZhpr(mcblasHandle_t handle, mcblasFillMode_t uplo,
                          int n, const double *alpha,
                          const mcDoubleComplex *x, int incx,
                          mcDoubleComplex *AP)
```

此函数执行压缩厄米秩一校正

$$A = \alpha \mathbf{x}\mathbf{x}^H + A$$

其中 A 是以压缩格式存储的 $n \times n$ 厄米矩阵，$\mathbf{x}$ 是向量，α 是标量。如果 uplo == MCBLAS_FILL_MODE_LOWER，厄米矩阵 A 下三角部分中的元素逐列压缩在一起，元素之间没有间隙，以便将元素 $A(i,j)$ 存储在 AP[i+((2*n-j+1)*j)/2]，其中 $j = 1,\ldots,n$，$i \geq j$。因此，压缩格式只需要 $\frac{n(n+1)}{2}$ 个元素进行存储。如果 uplo == MCBLAS_FILL_MODE_UPPER，厄米矩阵 A 上三角部分中的元素逐列压缩在一起，元素之间没有间隙，以便将元素 $A(i,j)$ 存储在 AP[i+(j*(j+1))/2]，其中 $j = 1,\ldots,n$，$i \leq j$。因此，压缩格式只需要 $\frac{n(n+1)}{2}$ 个元素进行存储。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
uplo		input	指示存储的是矩阵 A 的上半部分还是下半部分，另外 Hermitian 部分未被引用，且是从存储的元素推断出来的
n		input	矩阵 A 的行数和列数
alpha	主机或设备	input	用于乘法的 <type> 标量
x	设备	input	包含 n 个元素的 <type> 向量
incx		input	x 中连续元素之间的步幅
AP	设备	in/out	以压缩格式存储 A 的 <type> 数组假定对角线元素的虚数部分为 0

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_INVALID_VALUE	• n < 0 • incx = 0 • uplo != MCBLAS_FILL_MODE_UPPER, MCBLAS_FILL_MODE_LOWER • alpha = NULL
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.4.23 mcblas<t>hpr2()

```
mcblasStatus_t mcblasChpr2(mcblasHandle_t handle, mcblasFillMode_t uplo,
                           int n, const mcComplex       *alpha,
                           const mcComplex       *x, int incx,
                           const mcComplex       *y, int incy,
                           mcComplex       *AP)
mcblasStatus_t mcblasZhpr2(mcblasHandle_t handle, mcblasFillMode_t uplo,
                           int n, const mcDoubleComplex *alpha,
                           const mcDoubleComplex *x, int incx,
                           const mcDoubleComplex *y, int incy,
                           mcDoubleComplex *AP)
```

此函数执行压缩厄米秩二校正

$$A = \alpha \mathbf{x}\mathbf{y}^H + \overline{\alpha}\mathbf{y}\mathbf{x}^H + A$$

其中 A 是以压缩格式存储的 $n \times n$ 厄米矩阵，**x** 和 **y** 是向量，α 是标量。如果 uplo == MCBLAS_FILL_MODE_LOWER，厄米矩阵 A 下三角部分中的元素逐列压缩在一起，元素之间没有间隙，以便将元素 $A(i,j)$ 存储在 AP[i+((2*n-j+1)*j)/2]，其中 $j = 1,\ldots,n$，$i \geq j$。因此，压缩格式只需要 $\frac{n(n+1)}{2}$ 个元素进行存储。如果 uplo == MCBLAS_FILL_MODE_UPPER，厄米矩阵 A 上三角部分中的元素逐列压缩在一起，元素之间没有间隙，以便将元素 $A(i,j)$ 存储在 AP[i+(j*(j+1))/2]，其中 $j = 1,\ldots,n$，$i \leq j$。因此，压缩格式只需要 $\frac{n(n+1)}{2}$ 个元素进行存储。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
uplo		input	指示存储的是矩阵 A 的上半部分还是下半部分，另外 Hermitian 部分未被引用，且是从存储的元素推断出来的
n		input	矩阵 A 的行数和列数
alpha	主机或设备	input	用于乘法的 <type> 标量
x	设备	input	包含 n 个元素的 <type> 向量
incx		input	x 中连续元素之间的步幅
y	设备	input	包含 n 个元素的 <type> 向量
incy		input	y 中连续元素之间的步幅
AP	设备	in/out	以压缩格式存储 A 的 <type> 数组假定对角线元素的虚数部分为 0

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_INVALID_VALUE	• n < 0 • incx = 0 或 incy = 0 • uplo != MCBLAS_FILL_MODE_UPPER, MCBLAS_FILL_MODE_LOWER • alpha = NULL
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.4.24 mcblas<t>gemvBatched()

```
mcblasStatus_t mcblasSgemvBatched(mcblasHandle_t handle, mcblasOperation_t
↪trans,
                                  int m, int n,
                                  const float           *alpha,
                                  const float           *Aarray[], int lda,
                                  const float           *xarray[], int
↪incx,
                                  const float           *beta,
                                  float                 *yarray[], int
↪incy,
                                  int batchCount)
mcblasStatus_t mcblasDgemvBatched(mcblasHandle_t handle, mcblasOperation_t
↪trans,
                                  int m, int n,
                                  const double          *alpha,
                                  const double          *Aarray[], int lda,
                                  const double          *xarray[], int
↪incx,
                                  const double          *beta,
                                  double                *yarray[], int
↪incy,
                                  int batchCount)
mcblasStatus_t mcblasCgemvBatched(mcblasHandle_t handle, mcblasOperation_t
↪trans,
                                  int m, int n,
```

(下页继续)

(续上页)

```
                                const mcComplex       *alpha,
                                const mcComplex       *Aarray[], int lda,
                                const mcComplex       *xarray[], int
↪incx,
                                const mcComplex       *beta,
                                mcComplex             *yarray[], int
↪incy,
                                int batchCount)
mcblasStatus_t mcblasZgemvBatched(mcblasHandle_t handle, mcblasOperation_t
↪trans,
                                int m, int n,
                                const mcDoubleComplex *alpha,
                                const mcDoubleComplex *Aarray[], int lda,
                                const mcDoubleComplex *xarray[], int
↪incx,
                                const mcDoubleComplex *beta,
                                mcDoubleComplex       *yarray[], int
↪incy,
                                int batchCount)
mcblasStatus_t mcblasHSHgemvBatched(mcblasHandle_t handle, mcblasOperation_
↪t trans,
                                int m, int n,
                                const float           *alpha,
                                const mcblas_half     *Aarray[], int
↪lda,
                                const mcblas_half     *xarray[], int
↪incx,
                                const float           *beta,
                                mcblas_half           *yarray[], int
↪incy,
                                int batchCount)
mcblasStatus_t mcblasHSSgemvBatched(mcblasHandle_t handle, mcblasOperation_
↪t trans,
                                int m, int n,
                                const float           *alpha,
                                const mcblas_half     *Aarray[], int
↪lda,
                                const mcblas_half     *xarray[], int
↪incx,
                                const float           *beta,
                                float                 *yarray[], int
↪incy,
                                int batchCount)
mcblasStatus_t mcblasTSTgemvBatched(mcblasHandle_t handle, mcblasOperation_
↪t trans,
                                int m, int n,
                                const float           *alpha,
                                const mcblas_bfloat16 *Aarray[], int
↪lda,
                                const mcblas_bfloat16 *xarray[], int
↪incx,
                                const float           *beta,
                                mcblas_bfloat16       *yarray[], int
↪incy,
                                int batchCount)
mcblasStatus_t mcblasTSSgemvBatched(mcblasHandle_t handle, mcblasOperation_
↪t trans,
```

(下页继续)

(续上页)

```
                                int m, int n,
                                const float           *alpha,
                                const mcblas_bfloat16 *Aarray[], int
↪lda,
                                const mcblas_bfloat16 *xarray[], int
↪incx,
                                const float           *beta,
                                float                 *yarray[], int
↪incy,
                                int batchCount)
```

该函数对一批矩阵和向量执行矩阵-向量乘法。该批是“均匀”的，即所有实例对于各自的 A 矩阵，x 和 y 向量具有相同的维度（m，n），前导维度（lda），增量（incx，incy）和转置（trans）。批处理的每个实例的输入矩阵和向量的地址以及输出向量的地址都是从调用者传递给函数的指针数组中读取的。

$$y[i] = \alpha op(A[i])x[i] + \beta y[i],\ for\ i \in [0, batchCount - 1]$$

其中 α 和 β 是标量，并且 A 是一个指针数组，指向矩阵 $A[i]$），**x** 和 **y** 是指向向量的指针数组。此外，对于矩阵 $A[i]$，

$$op(A[i]) = \begin{cases} A[i] & if\ trans == MCBLAS\_OP\_N \\ A[i]^T & if\ trans == MCBLAS\_OP\_T \\ A[i]^H & if\ trans == MCBLAS\_OP\_C \end{cases}$$

注解： **y**$[i]$ 向量不能重叠，即单个 gemv 操作必须独立计算；否则，会导致未定义行为。在某些 problem size 上，可能需要在不同的 MXMACA 流中多次调用 `mcblas<t>gemv`，而不是使用此 API。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
trans		input	非转置或共轭转置运算 op(`A[i]`)
m		input	矩阵 `A[i]` 的行数
n		input	矩阵 `A[i]` 的列数
alpha	主机或设备	input	用于乘法的 <type> 标量
Aarray	设备	input	指向 <type> 数组的指针数组，每个数组维度为 `lda x n`，其中 `lda >=max(1,m)`。 所有指针都必须满足特定的对齐条件。详情参见以下内容
lda		input	用于存储矩阵 `A[i]` 的二维数组的前导维度
xarray	设备	input	指向 <type> 数组的指针数组，如果 `trans == MCBLAS_OP_N`，数组维度为 `n`；否则为 `m`。 所有指针都必须满足特定的对齐条件。详情参见以下内容
incx		input	每个一维数组 `x[i]` 的步幅
beta	主机或设备	input	用于乘法的 <type> 标量，如果 `beta == 0`，则 `y` 不必为有效输入
yarray	设备	in/out	指向 <type> 数组的指针数组。如果 `trans == MCBLAS_OP_N`，数组维度为 `n`；否则为 `m`。向量 `y[i]` 不应重叠；否则，将出现未定义的行为。 所有指针都必须满足特定的对齐条件。详情参见以下内容
incy		input	每个一维数组 `y[i]` 的步幅
batchCount		input	Aarray、xarray 和 yarray 中包含的指针数

如果使用 `mcblasSgemvBatched()` 时，数学模式启用快速数学模式，则 GPU 内存中的指针（而非指针数组）必须正确对齐，以避免未对齐的内存访问错误。理想情况下，所有指针都至少对齐 16 个字节。否则，建议符合以下规则：

- 如果 `k % 4==0`，则确保 `intptr_t(ptr) % 16 == 0`

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_INVALID_VALUE	参数 m,n,batchCount<0
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.4.25 mcblas<t>gemvStridedBatched()

```
mcblasStatus_t mcblasSgemvStridedBatched(mcblasHandle_t handle,
                                         mcblasOperation_t trans,
                                         int m, int n,
                                         const float           *alpha,
                                         const float           *A, int lda,
                                         mcblas_stride         strideA,
                                         const float           *x, int
 ↪incx,
                                         mcblas_stride         stridex,
                                         const float           *beta,
                                         float                 *y, int
 ↪incy,
                                         mcblas_stride         stridey,
                                         int batchCount)
mcblasStatus_t mcblasDgemvStridedBatched(mcblasHandle_t handle,
                                         mcblasOperation_t trans,
                                         int m, int n,
                                         const double          *alpha,
                                         const double          *A, int lda,
                                         mcblas_stride         strideA,
                                         const double          *x, int
 ↪incx,
                                         mcblas_stride         stridex,
                                         const double          *beta,
                                         double                *yarray[],
 ↪int incy,
                                         mcblas_stride         stridey,
                                         int batchCount)
mcblasStatus_t mcblasCgemvStridedBatched(mcblasHandle_t handle,
                                         mcblasOperation_t trans,
                                         int m, int n,
                                         const mcComplex       *alpha,
                                         const mcComplex       *A, int lda,
                                         mcblas_stride         strideA,
                                         const mcComplex       *x, int
 ↪incx,
                                         mcblas_stride         stridex,
                                         const mcComplex       *beta,
                                         mcComplex             *y, int
 ↪incy,
                                         mcblas_stride         stridey,
                                         int batchCount)
mcblasStatus_t mcblasZgemvStridedBatched(mcblasHandle_t handle,
                                         mcblasOperation_t trans,
```

(下页继续)

(续上页)

```
                                        int m, int n,
                                        const mcDoubleComplex *alpha,
                                        const mcDoubleComplex *A, int lda,
                                        mcblas_stride         strideA,
                                        const mcDoubleComplex *x, int
 ↪incx,
                                        mcblas_stride         stridex,
                                        const mcDoubleComplex *beta,
                                        mcDoubleComplex       *y, int
 ↪incy,
                                        mcblas_stride         stridey,
                                        int batchCount)
mcblasStatus_t mcblasHSHgemvStridedBatched(mcblasHandle_t handle,
                                          mcblasOperation_t trans,
                                          int m, int n,
                                          const float          *alpha,
                                          const mcblas_half    *A, int lda,
                                          mcblas_stride        strideA,
                                          const mcblas_half    *x, int
 ↪incx,
                                          mcblas_stride        stridex,
                                          const float          *beta,
                                          mcblas_half          *y, int
 ↪incy,
                                          mcblas_stride        stridey,
                                          int batchCount)
mcblasStatus_t mcblasHSSgemvStridedBatched(mcblasHandle_t handle,
                                          mcblasOperation_t trans,
                                          int m, int n,
                                          const float          *alpha,
                                          const mcblas_half    *A, int lda,
                                          mcblas_stride        strideA,
                                          const mcblas_half    *x, int
 ↪incx,
                                          mcblas_stride        stridex,
                                          const float          *beta,
                                          float                *y, int
 ↪incy,
                                          mcblas_stride        stridey,
                                          int batchCount)
mcblasStatus_t mcblasTSTgemvStridedBatched(mcblasHandle_t handle,
                                          mcblasOperation_t trans,
                                          int m, int n,
                                          const float          *alpha,
                                          const mc_bfloat16    *A, int lda,
                                          mcblas_stride        strideA,
                                          const mc_bfloat16    *x, int
 ↪incx,
                                          mcblas_stride        stridex,
                                          const float          *beta,
                                          mc_bfloat16          *y, int
 ↪incy,
                                          mcblas_stride        stridey,
                                          int batchCount)
mcblasStatus_t mcblasTSSgemvStridedBatched(mcblasHandle_t handle,
                                          mcblasOperation_t trans,
```

(下页继续)

(续上页)

```
                                        int m, int n,
                                        const float          *alpha,
                                        const mc_bfloat16    *A, int lda,
                                        mcblas_stride        strideA,
                                        const mc_bfloat16    *x, int
↪incx,
                                        mcblas_stride        stridex,
                                        const float          *beta,
                                        float                *y, int
↪incy,
                                        mcblas_stride        stridey,
                                        int batchCount)
```

该函数对一批矩阵和向量执行矩阵-向量乘法。该批是“均匀”的，即所有实例对于各自的 A 矩阵，x 和 y 向量具有相同的维度（m，n），前导维度（lda），增量（incx，incy）和转置（trans）。批处理中每个实例的输入矩阵 A 和向量 x 以及输出向量 y 的位置，与前一个实例中的位置在元素数量上有固定的偏移。第一个实例中指向 A 矩阵，x 和 y 向量的指针，与元素数量的偏移量（strideA、stridex、stridey）一并由用户传入函数，这些偏移量决定了后续实例中输入矩阵和向量以及输出向量的位置。

$$\mathbf{y} + i * stridey = \alpha \text{op}(A + i * strideA)(\mathbf{x} + i * stridex) + \beta(\mathbf{y} + i * stridey),$$
$$\text{for i} \in [0, batchCount - 1]$$

其中 α 和 β 是标量，并且 A 是一个指针数组，指向矩阵 $A[i]$），**x** 和 **y** 是指向向量的指针数组。此外，对于矩阵 $A[i]$

$$op(A[i]) = \begin{cases} A[i] & if\ \ trans == MCBLAS\_OP\_N \\ A[i]^T & if\ \ trans == MCBLAS\_OP\_T \\ A[i]^H & if\ \ trans == MCBLAS\_OP\_C \end{cases}$$

注解： **y**$[i]$ 矩阵不能重叠，即单个 gemv 操作必须独立计算；否则，会导致未定义行为。在某些 problem size 上，可能需要在不同的 MXMACA 流中多次调用 `mcblas<t>gemv`，而不是使用此 API。

在下表中，我们使用 `A[i], x[i], y[i]` 分别表示 A、B、C 矩阵，以及批处理中第 i 个实例中的 x 和 y 向量，隐式假设它们分别是相对于 `A[i-1], x[i-1], y[i-1]` 的元素数量的偏移 `strideA, stridex, stridey`。偏移的单位是元素数量，不能为零。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
trans		input	非转置或共轭转置运算 op(`A[i]`)
m		input	矩阵 `A[i]` 的行数
n		input	矩阵 `A[i]` 的列数
alpha	主机或设备	input	用于乘法的 <type> 标量
A	设备	input	指向批处理第一个实例对应的矩阵 `A` 的 <type>* 指针，维度为 `lda x n`，其中 `lda >=max(1,m)`
lda		input	用于存储矩阵 `A[i]` 的二维数组的前导维度
strideA		input	long long int 类型的值，该值给出 `A[i]` 和 `A[i+1]` 之间的元素数偏移量
x	设备	input	指向批处理第一个实例对应的 `x` 向量的 <type>* 指针，如果 `trans==MCBLAS_OP_N`，维度为 `n`，否则，为 `m`
incx		input	每个一维数组 `x[i]` 的步幅
stridex		input	long long int 类型的值，该值给出 `x[i]` 和 `x[i+1]` 之间的元素数偏移量
beta	主机或设备	input	用于乘法的 <type> 标量，如果 `beta == 0`，则 `y` 不必为有效输入

下页继续

表 2.2 – 续上页

参数	内存	In/out	含义
y	设备	in/out	指向批处理第一个实例对应的 y 向量的 <type>* 指针，如果 trans==MCBLAS_OP_N，维度为 m，否则，为 n。向量 y[i] 不应重叠；否则，将出现未定义的行为。
incy		input	每个一维数组 y[i] 的步幅
stridey		input	long long int 类型的值，该值给出 y[i] 和 y[i+1] 之间的元素数偏移量
batchCount		input	Aarray、xarray 和 yarray 中包含的指针数

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_INVALID_VALUE	参数 m,n,batchCount<0
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.5 mcBLAS Level-3 函数

本章中，我们将介绍执行矩阵-矩阵操作的 Level-3 基本线性代数子程序（BLAS3）函数。

2.5.1 mcblas<t>gemm()

```
mcblasStatus_t mcblasSgemm(mcblasHandle_t handle,
                           mcblasOperation_t transa, mcblasOperation_t
↪transb,
                           int m, int n, int k,
                           const float           *alpha,
                           const float           *A, int lda,
                           const float           *B, int ldb,
                           const float           *beta,
                           float                 *C, int ldc)
mcblasStatus_t mcblasDgemm(mcblasHandle_t handle,
                           mcblasOperation_t transa, mcblasOperation_t
↪transb,
                           int m, int n, int k,
                           const double          *alpha,
                           const double          *A, int lda,
                           const double          *B, int ldb,
                           const double          *beta,
                           double                *C, int ldc)
mcblasStatus_t mcblasCgemm(mcblasHandle_t handle,
                           mcblasOperation_t transa, mcblasOperation_t
↪transb,
                           int m, int n, int k,
                           const mcComplex       *alpha,
                           const mcComplex       *A, int lda,
                           const mcComplex       *B, int ldb,
                           const mcComplex       *beta,
                           mcComplex             *C, int ldc)
mcblasStatus_t mcblasZgemm(mcblasHandle_t handle,
```

(下页继续)

(续上页)

```
                            mcblasOperation_t transa, mcblasOperation_t
↪transb,
                            int m, int n, int k,
                            const mcDoubleComplex *alpha,
                            const mcDoubleComplex *A, int lda,
                            const mcDoubleComplex *B, int ldb,
                            const mcDoubleComplex *beta,
                            mcDoubleComplex       *C, int ldc)
mcblasStatus_t mcblasHgemm(mcblasHandle_t handle,
                            mcblasOperation_t transa, mcblasOperation_t
↪transb,
                            int m, int n, int k,
                            const mcblas_half     *alpha,
                            const mcblas_half     *A, int lda,
                            const mcblas_half     *B, int ldb,
                            const mcblas_half     *beta,
                            mcblas_half           *C, int ldc)
```

该函数执行矩阵-矩阵乘法

$$C = \alpha \text{op}(A)\text{op}(B) + \beta C$$

其中 α 和 β 是标量，A、B、C 是以列主格式存储的矩阵，维度分别为 op(A) $m \times k$，op(B) $k \times n$ 和 C $m \times n$。此外，对于矩阵 A，

$$op(A) = \begin{cases} A & if\ \ transa == MCBLAS\_OP\_N \\ A^T & if\ \ transa == MCBLAS\_OP\_T \\ A^H & if\ \ transa == MCBLAS\_OP\_C \end{cases}$$

对于矩阵 B，op(B) 的定义类似。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
transa		input	非转置或共轭转置运算 op(A)
transb		input	非转置或共轭转置运算 op(B)
m		input	矩阵 op(A) 和 C 的行数
n		input	矩阵 op(B) 和 C 的列数
k		input	op(A) 的列数和 op(B) 的行数
alpha	主机或设备	input	用于乘法的 <type> 标量
A	设备	input	如果 `transa == MCBLAS_OP_N`，<type> 数组维度为 `lda x k`，其中 `lda>=max(1,m)`；否则维度为 `lda x m`，其中 `lda>=max(1,k)`
lda		input	用于存储矩阵 A 的二维数组的前导维度
B	设备	input	如果 `transb == MCBLAS_OP_N`，<type> 数组维度为 `lda x n`，其中 `lda>=max(1,k)`；否则维度为 `lda x k`，其中 `lda>=max(1,n)`
ldb		input	用于存储矩阵 B 的二维数组的前导维度
beta	主机或设备	input	用于乘法的 <type> 标量，如果 `beta == 0`，则 C 不必为有效输入
C	设备	in/out	<type> 数组，维度为 `ldc x n`，其中 `ldc>=max(1,m)`
ldc		input	用于存储矩阵 C 的二维数组的前导维度

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_INVALID_VALUE	• m, n, k < 0 • transa, transb != MCBLAS_OP_N, MCBLAS_OP_C, MCBLAS_OP_T • 如果 transa == MCBLAS_OP_N，lda < max(1, m)；否则 lda < max(1, k) • 如果 transb == MCBLAS_OP_N，ldb < max(1, k)；否则 ldb < max(1, n) • ldc < max(1, m) • alpha, beta == NULL • 如果 C 需要缩放，C = NULL
MCBLAS_STATUS_ARCH_MISMATCH	在 mcblasHgemm 的情况下，该设备不支持半精度的数学运算
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.5.2 mcblas<t>gemm3m()

```
mcblasStatus_t mcblasCgemm3m(mcblasHandle_t handle,
                             mcblasOperation_t transa, mcblasOperation_t
↪transb,
                             int m, int n, int k,
                             const mcComplex       *alpha,
                             const mcComplex       *A, int lda,
                             const mcComplex       *B, int ldb,
                             const mcComplex       *beta,
                             mcComplex             *C, int ldc)
mcblasStatus_t mcblasZgemm3m(mcblasHandle_t handle,
                             mcblasOperation_t transa, mcblasOperation_t
↪transb,
                             int m, int n, int k,
                             const mcDoubleComplex *alpha,
                             const mcDoubleComplex *A, int lda,
                             const mcDoubleComplex *B, int ldb,
                             const mcDoubleComplex *beta,
                             mcDoubleComplex       *C, int ldc)
```

该函数使用高斯复杂度归约（Gauss complexity reduction）算法执行复矩阵-矩阵乘法。这可使性能提升高达 25%。

$$C = \alpha \text{op}(A)\text{op}(B) + \beta C$$

其中 α 和 β 是标量，A、B、C 是以列主格式存储的矩阵，维度分别为 op(A) $m \times k$，op(B) $k \times n$ 和 C $m \times n$。此外，对于矩阵 A，

$$op(A) = \begin{cases} A & if\ transa == MCBLAS\_OP\_N \\ A^T & if\ transa == MCBLAS\_OP\_T \\ A^H & if\ transa == MCBLAS\_OP\_C \end{cases}$$

对于矩阵 B，op(B) 的定义类似。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
transa		input	非转置或共轭转置运算 op(A)
transb		input	非转置或共轭转置运算 op(B)
m		input	矩阵 op(A) 和 C 的行数
n		input	矩阵 op(B) 和 C 的列数
k		input	op(A) 的列数和 op(B) 的行数
alpha	主机或设备	input	用于乘法的 <type> 标量
A	设备	input	如果 transa == MCBLAS_OP_N，<type> 数组维度为 lda x k，其中 lda>=max(1,m)；否则维度为 lda x m，其中 lda>=max(1,k)
lda		input	用于存储矩阵 A 的二维数组的前导维度
B	设备	input	如果 transb == MCBLAS_OP_N，<type> 数组维度为 lda x n，其中 lda>=max(1,k)；否则维度为 lda x k，其中 lda>=max(1,n)
ldb		input	用于存储矩阵 B 的二维数组的前导维度
beta	主机或设备	input	用于乘法的 <type> 标量，如果 beta == 0，则 C 不必为有效输入
C	设备	in/out	<type> 数组，维度为 ldc x n，其中 ldc>=max(1,m)
ldc		input	用于存储矩阵 C 的二维数组的前导维度

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_INVALID_VALUE	• m, n, k < 0 • transa, transb != MCBLAS_OP_N, MCBLAS_OP_C, MCBLAS_OP_T • 如果 transa == MCBLAS_OP_N，lda < max(1, m)；否则 lda < max(1, k) • 如果 transb == MCBLAS_OP_N，ldb < max(1, k)；否则 ldb < max(1, n) • ldc < max(1, m) • alpha, beta == NULL • 如果 C 需要缩放，C = NULL
MCBLAS_STATUS_ARCH_MISMATCH	不支持该设备
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.5.3 mcblas<t>gemmBatched()

```
mcblasStatus_t mcblasHgemmBatched(mcblasHandle_t handle,
                                  mcblasOperation_t transa,
                                  mcblasOperation_t transb,
                                  int m, int n, int k,
                                  const mcblas_half           *alpha,
                                  const mcblas_half           *Aarray[],
↪int lda,
                                  const mcblas_half           *Barray[],
↪int ldb,
                                  const mcblas_half           *beta,
                                  mcblas_half           *Carray[], int ldc,
```

(下页继续)

(续上页)

```
                                    int batchCount)
mcblasStatus_t mcblasSgemmBatched(mcblasHandle_t handle,
                                    mcblasOperation_t transa,
                                    mcblasOperation_t transb,
                                    int m, int n, int k,
                                    const float           *alpha,
                                    const float           *Aarray[], int lda,
                                    const float           *Barray[], int ldb,
                                    const float           *beta,
                                    float           *Carray[], int ldc,
                                    int batchCount)
mcblasStatus_t mcblasDgemmBatched(mcblasHandle_t handle,
                                    mcblasOperation_t transa,
                                    mcblasOperation_t transb,
                                    int m, int n, int k,
                                    const double          *alpha,
                                    const double          *Aarray[], int lda,
                                    const double          *Barray[], int ldb,
                                    const double          *beta,
                                    double          *Carray[], int ldc,
                                    int batchCount)
mcblasStatus_t mcblasCgemmBatched(mcblasHandle_t handle,
                                    mcblasOperation_t transa,
                                    mcblasOperation_t transb,
                                    int m, int n, int k,
                                    const mcComplex       *alpha,
                                    const mcComplex       *Aarray[], int lda,
                                    const mcComplex       *Barray[], int ldb,
                                    const mcComplex       *beta,
                                    mcComplex       *Carray[], int ldc,
                                    int batchCount)
mcblasStatus_t mcblasZgemmBatched(mcblasHandle_t handle,
                                    mcblasOperation_t transa,
                                    mcblasOperation_t transb,
                                    int m, int n, int k,
                                    const mcDoubleComplex *alpha,
                                    const mcDoubleComplex *Aarray[], int lda,
                                    const mcDoubleComplex *Barray[], int ldb,
                                    const mcDoubleComplex *beta,
                                    mcDoubleComplex *Carray[], int ldc,
                                    int batchCount)
```

该函数对一批矩阵和向量执行矩阵-矩阵乘法。该批是“均匀”的，即所有实例对于各自的 A、B、C 矩阵具有相同的维度（m，n，k），前导维度（lda，ldb，ldc）和转置（transa，transb）。批处理的每个实例的输入矩阵和输出矩阵的地址都是从调用者传递给函数的指针数组中读取的。

$$C[i] = \alpha \text{op}(A[i])\text{op}(B[i]) + \beta C[i],$$
$$\text{for i} \in [0, batchCount - 1]$$

其中 α 和 β 是标量，A、B、C 是指针数组，指向以列主格式存储的矩阵，维度分别为 op$(A[i])$ $m \times k$，op$(B[i])$ $k \times n$ 和 $C[i]$ $m \times n$。此外，对于矩阵 $A[i]$，

$$op(A) = \begin{cases} A & if\ \ transa == MCBLAS\_OP\_N \\ A^T & if\ \ transa == MCBLAS\_OP\_T \\ A^H & if\ \ transa == MCBLAS\_OP\_C \end{cases}$$

对于矩阵 $B[i]$，op$(B[i])$ 的定义类似。

注解: $C[i]$ 矩阵不能重叠，即单个 gemm 操作必须独立计算；否则，会导致未定义行为。

在某些 problem size 上，可能需要在不同的 MXMACA 流中多次调用 `mcblas<t>gemm`，而不是使用此 API。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
transa		input	非转置或共轭转置运算 op(`A[i]`)
transb		input	非转置或共轭转置运算 op(`B[i]`)
m		input	矩阵 op(`A[i]`) 和 `C[i]` 的行数
n		input	矩阵 op(`B[i]`) 和 `C[i]` 的列数
k		input	op(`A[i]`) 的列数和 op(`B[i]`) 的行数
alpha	主机或设备	input	用于乘法的 <type> 标量
Aarray	设备	input	指向 <type> 数组的指针数组，如果 `transa == MCBLAS_OP_N`，数组维度为 `lda x k`，其中 `lda>=max(1,m)`；否则，数组维度为 `lda x m`，其中 `lda>=max(1,k)`。 所有指针都必须满足特定的对齐条件。详情参见以下内容
lda		input	用于存储矩阵 `A[i]` 的二维数组的前导维度
Barray	设备	input	指向 <type> 数组的指针数组，如果 `transb == MCBLAS_OP_N`，数组维度为 `lda x n`，其中 `lda>=max(1,k)`；否则，数组维度为 `lda x k`，其中 `lda>=max(1,n)`。 所有指针都必须满足特定的对齐条件。详情参见以下内容
ldb		input	用于存储矩阵 `B[i]` 的二维数组的前导维度
beta	主机或	input	用于乘法的 <type> 标量，如果 `beta == 0`，则 `C` 不必为有效输入
Carray	设备	in/out	指向 <type> 数组的指针数组。维度为 `ldc x n`，其中 `ldc>=max(1,m)`。矩阵 `C[i]` 不应重叠；否则，将出现未定义的行为。 所有指针都必须满足特定的对齐条件。详情参见以下内容
ldc		input	用于存储矩阵 `C[i]` 的二维数组的前导维度
batchCount		input	Aarray、Barray 和 Carray 中包含的指针数

如果使用 `mcblasSgemmBatched()` 时，数学模式启用快速数学模式，则 GPU 内存中的指针（而非指针数组）必须正确对齐，以避免未对齐的内存访问错误。理想情况下，所有指针都至少对齐 16 个字节。否则，建议符合以下规则：

- 如果 `k%4==0`，则确保 `intptr_t(ptr) % 16 == 0`

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_INVALID_VALUE	• m, n, k, batchCount < 0 • transa, transb != MCBLAS_OP_N, MCBLAS_OP_C, MCBLAS_OP_T • 如果transa == MCBLAS_OP_N,lda < max(1, m)；否则lda < max(1, k) • 如果transb == MCBLAS_OP_N,ldb < max(1, k)；否则ldb < max(1, n) • ldc < max(1, m)
MCBLAS_STATUS_ARCH_MISMATCH	不支持该设备
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.5.4 mcblas<t>gemmStridedBatched()

```
mcblasStatus_t mcblasHgemmStridedBatched(mcblasHandle_t handle,
                                         mcblasOperation_t transa,
                                         mcblasOperation_t transb,
                                         int m, int n, int k,
                                         const mcblas_half     *alpha,
                                         const mcblas_half     *A, int lda,
                                         mcblas_stride          strideA,
                                         const mcblas_half     *B, int ldb,
                                         mcblas_stride          strideB,
                                         const mcblas_half     *beta,
                                         mcblas_half           *C, int ldc,
                                         mcblas_stride          strideC,
                                         int batchCount)
mcblasStatus_t mcblasSgemmStridedBatched(mcblasHandle_t handle,
                                         mcblasOperation_t transa,
                                         mcblasOperation_t transb,
                                         int m, int n, int k,
                                         const float           *alpha,
                                         const float           *A, int lda,
                                         mcblas_stride          strideA,
                                         const float           *B, int ldb,
                                         mcblas_stride          strideB,
                                         const float           *beta,
                                         float                 *C, int ldc,
                                         mcblas_stride          strideC,
                                         int batchCount)
mcblasStatus_t mcblasDgemmStridedBatched(mcblasHandle_t handle,
                                         mcblasOperation_t transa,
                                         mcblasOperation_t transb,
                                         int m, int n, int k,
                                         const double          *alpha,
                                         const double          *A, int lda,
                                         mcblas_stride          strideA,
                                         const double          *B, int ldb,
                                         mcblas_stride          strideB,
                                         const double          *beta,
                                         double                *C, int ldc,
```

(下页继续)

(续上页)

```
                                        mcblas_stride          strideC,
                                        int batchCount)
mcblasStatus_t mcblasCgemmStridedBatched(mcblasHandle_t handle,
                                        mcblasOperation_t transa,
                                        mcblasOperation_t transb,
                                        int m, int n, int k,
                                        const mcComplex       *alpha,
                                        const mcComplex       *A, int lda,
                                        mcblas_stride          strideA,
                                        const mcComplex       *B, int ldb,
                                        mcblas_stride          strideB,
                                        const mcComplex       *beta,
                                        mcComplex             *C, int ldc,
                                        mcblas_stride          strideC,
                                        int batchCount)
mcblasStatus_t mcblasCgemm3mStridedBatched(mcblasHandle_t handle,
                                          mcblasOperation_t transa,
                                          mcblasOperation_t transb,
                                          int m, int n, int k,
                                          const mcComplex      *alpha,
                                          const mcComplex      *A, int
 ↪lda,
                                          mcblas_stride         strideA,
                                          const mcComplex      *B, int
 ↪ldb,
                                          mcblas_stride         strideB,
                                          const mcComplex      *beta,
                                          mcComplex            *C, int
 ↪ldc,
                                          mcblas_stride         strideC,
                                          int batchCount)
mcblasStatus_t mcblasZgemmStridedBatched(mcblasHandle_t handle,
                                        mcblasOperation_t transa,
                                        mcblasOperation_t transb,
                                        int m, int n, int k,
                                        const mcDoubleComplex *alpha,
                                        const mcDoubleComplex *A, int lda,
                                        mcblas_stride          strideA,
                                        const mcDoubleComplex *B, int ldb,
                                        mcblas_stride          strideB,
                                        const mcDoubleComplex *beta,
                                        mcDoubleComplex       *C, int ldc,
                                        mcblas_stride          strideC,
                                        int batchCount)
```

该函数对一批矩阵和向量执行矩阵-矩阵乘法。该批是“均匀”的，即所有实例对于各自的 A、B、C 矩阵具有相同的维度（m，n，k），前导维度（lda，ldb，ldc）和转置（transa，transb）。批处理中每个实例的输入矩阵 A 和 B 以及输出矩阵 C 的位置，与前一个实例中的位置在元素数量上有固定的偏移。第一个实例中指向矩阵 A、B、C 的指针，与元素数量的偏移量（strideA、strideB、strideC）一并由用户传入函数，这些偏移量决定了后续实例中输入矩阵和输出矩阵的位置。

$$C + i * strideC = \alpha \text{op}(A + i * strideA)\text{op}(B + i * strideB) + \beta(C + i * strideC),$$
$$\text{for i} \in [0, batchCount - 1]$$

其中 α 和 β 是标量，A、B、C 是指针数组，指向以列主格式存储的矩阵，维度分别为 $\text{op}(A[i])$ $m \times k$，

op($B[i]$) $k \times n$ 和 $C[i]$ $m \times n$。此外，对于矩阵 $A[i]$，

$$op(A) = \begin{cases} A & if\ transa == MCBLAS\_OP\_N \\ A^T & if\ transa == MCBLAS\_OP\_T \\ A^H & if\ transa == MCBLAS\_OP\_C \end{cases}$$

对于矩阵 $B[i]$，op($B[i]$) 的定义类似。

注解： $C[i]$ 矩阵不能重叠，即单个 gemm 操作必须独立计算；否则，会导致未定义行为。

在某些 problem size 上，可能需要在不同的 mcStream 中多次调用 `mcblas<t>gemm`，而不是使用此 API。

在下表中，我们使用 `A[i], B[i], C[i]` 分别表示批处理的第 i 个实例中 A、B、C 矩阵，隐式假设它们分别是相对于 `A[i-1], B[i-1], C[i-1]` 的元素数量的偏移 `strideA, strideB, strideC`。偏移的单位是元素数量，不能为零。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
transa		input	非转置或共轭转置运算 op(`A[i]`)
transb		input	非转置或共轭转置运算 op(`B[i]`)
m		input	矩阵 op(`A[i]`) 和 `C[i]` 的行数
n		input	矩阵 op(`B[i]`) 和 `C[i]` 的列数
k		input	op(`A[i]`) 的列数和 op(`B[i]`) 的行数
alpha	主机或设备	input	用于乘法的 <type> 标量
A	设备	input	指向批处理第一个实例对应的矩阵 `A` 的 <type>* 指针，如果 `transa==MCBLAS_OP_N`，维度为 `lda x k`，其中 `lda>=max(1,m)`；否则，维度为 `lda x m`，其中 `lda>=max(1,k)`
lda		input	用于存储矩阵 `A[i]` 的二维数组的前导维度
strideA		input	long long int 类型的值，该值给出 `A[i]` 和 `A[i+1]` 之间的元素数偏移量
B	设备	input	指向批处理第一个实例对应的矩阵 `B` 的 <type>* 指针，如果 `transb==MCBLAS_OP_N`，维度为 `ldb x n`，其中 `ldb>=max(1,k)`；否则，维度为 `ldb x k`，其中 `ldb>=max(1,n)`
ldb		input	用于存储矩阵 `B[i]` 的二维数组的前导维度
strideB		input	long long int 类型的值，该值给出 `B[i]` 和 `B[i+1]` 之间的元素数偏移量
beta	主机或	input	用于乘法的 <type> 标量，如果 `beta == 0`，则 `C` 不必为有效输入
C	设备	in/out	指向批处理第一个实例对应的矩阵 `C` 的 <type>* 指针，维度为 `ldc x n`，其中 `ldc>=max(1,m)`。矩阵 `C[i]` 不应重叠；否则，将出现未定义的行为。
ldc		input	用于存储矩阵 `C[i]` 的二维数组的前导维度
strideC		input	long long int 类型的值，该值给出 `C[i]` 和 `C[i+1]` 之间的元素数偏移量
batchCount		input	批处理中要执行的 GEMM 数量

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_INVALID_VALUE	• m, n, k, batchCount < 0 • transa, transb != MCBLAS_OP_N, MCBLAS_OP_C, MCBLAS_OP_T • 如果transa == MCBLAS_OP_N,lda < max(1, m)；否则lda < max(1, k) • 如果transb == MCBLAS_OP_N,ldb < max(1, k)；否则ldb < max(1, n) • ldc < max(1, m)
MCBLAS_STATUS_ARCH_MISMATCH	不支持该设备
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.5.5 mcblas<t>symm()

```
mcblasStatus_t mcblasSsymm(mcblasHandle_t handle,
                           mcblasSideMode_t side, mcblasFillMode_t uplo,
                           int m, int n,
                           const float           *alpha,
                           const float           *A, int lda,
                           const float           *B, int ldb,
                           const float           *beta,
                           float                 *C, int ldc)
mcblasStatus_t mcblasDsymm(mcblasHandle_t handle,
                           mcblasSideMode_t side, mcblasFillMode_t uplo,
                           int m, int n,
                           const double          *alpha,
                           const double          *A, int lda,
                           const double          *B, int ldb,
                           const double          *beta,
                           double                *C, int ldc)
mcblasStatus_t mcblasCsymm(mcblasHandle_t handle,
                           mcblasSideMode_t side, mcblasFillMode_t uplo,
                           int m, int n,
                           const mcComplex       *alpha,
                           const mcComplex       *A, int lda,
                           const mcComplex       *B, int ldb,
                           const mcComplex       *beta,
                           mcComplex             *C, int ldc)
mcblasStatus_t mcblasZsymm(mcblasHandle_t handle,
                           mcblasSideMode_t side, mcblasFillMode_t uplo,
                           int m, int n,
                           const mcDoubleComplex *alpha,
                           const mcDoubleComplex *A, int lda,
                           const mcDoubleComplex *B, int ldb,
                           const mcDoubleComplex *beta,
                           mcDoubleComplex       *C, int ldc)
```

该函数执行对称矩阵-矩阵乘法

$$C = \begin{cases} \alpha AB + \beta C & if\ \ side == MCBLAS\_SIDE\_LEFT \\ \alpha BA + \beta C & if\ \ side == MCBLAS\_SIDE\_RIGHT \end{cases}$$

其中 A 是以 lower 或 upper 模式存储的对称矩阵，B 和 C 是 $m \times n$ 矩阵，α 和 β 是标量。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
side		input	指示矩阵 A 是在 B 的左边还是右边
uplo		input	指示存储的是矩阵 A 的上半部分还是下半部分，其对称部分未被引用，且是从存储的元素推断出来的
m		input	矩阵 C 和 B 的行数，矩阵 A 的大小也相应调整
n		input	矩阵 C 和 B 的列数，矩阵 A 的大小也相应调整
alpha	主机或设备	input	用于乘法的 <type> 标量
A	设备	input	如果 side == MCBLAS_SIDE_LEFT，<type> 数组维度为 lda x m，其中 lda>=max(1,m)；否则，维度为 lda x n，其中 lda>=max(1,n)
lda		input	用于存储矩阵 A 的二维数组的前导维度
B	设备	input	<type> 数组，维度为 ldb x n，其中 ldb>=max(1,m)
ldb		input	用于存储矩阵 B 的二维数组的前导维度
beta	主机或设备	input	用于乘法的 <type> 标量，如果 beta == 0，则 C 不必为有效输入
C	设备	in/out	<type> 数组，维度为 ldc x n，其中 ldc>=max(1,m)
ldc		input	用于存储矩阵 C 的二维数组的前导维度

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_INVALID_VALUE	• m, n < 0 • side != MCBLAS_SIDE_LEFT, MCBLAS_SIDE_RIGHT • uplo != MCBLAS_FILL_MODE_LOWER, MCBLAS_FILL_MODE_UPPER • 如果 side == MCBLAS_SIDE_LEFT ，lda < max(1, m)；否则 lda < max(1, n) • ldb < max(1, m) • ldc < max(1, m) • alpha == NULL 或 beta == NULL • 如果 C 需要缩放，C = NULL
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.5.6 mcblas<t>syrk()

```
mcblasStatus_t mcblasSsyrk(mcblasHandle_t handle,
                           mcblasFillMode_t uplo, mcblasOperation_t trans,
                           int n, int k,
                           const float           *alpha,
                           const float           *A, int lda,
                           const float           *beta,
                           float           *C, int ldc)
mcblasStatus_t mcblasDsyrk(mcblasHandle_t handle,
                           mcblasFillMode_t uplo, mcblasOperation_t trans,
                           int n, int k,
                           const double          *alpha,
                           const double          *A, int lda,
                           const double          *beta,
```

(下页继续)

(续上页)

```
                            double          *C, int ldc)
mcblasStatus_t mcblasCsyrk(mcblasHandle_t handle,
                            mcblasFillMode_t uplo, mcblasOperation_t trans,
                            int n, int k,
                            const mcComplex       *alpha,
                            const mcComplex       *A, int lda,
                            const mcComplex       *beta,
                            mcComplex       *C, int ldc)
mcblasStatus_t mcblasZsyrk(mcblasHandle_t handle,
                            mcblasFillMode_t uplo, mcblasOperation_t trans,
                            int n, int k,
                            const mcDoubleComplex *alpha,
                            const mcDoubleComplex *A, int lda,
                            const mcDoubleComplex *beta,
                            mcDoubleComplex *C, int ldc)
```

此函数执行对称秩- k 校正 $C = \alpha \text{op}(A) \text{op}(A)^T + \beta C$，其中 α 和 β 是标量，C 是以 lower 或 upper 模式存储的对称矩阵，A 是一个维度为 op(A) $n \times k$ 的矩阵。此外，对于矩阵 A

$$op(A) = \begin{cases} A & if\ transa == MCBLAS\_OP\_N \\ A^T & if\ transa == MCBLAS\_OP\_T \end{cases}$$

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
uplo		input	指示存储的是矩阵 C 的上半部分还是下半部分，其对称部分未被引用，且是从存储的元素推断出来的
trans		input	非转置或运算 op(A)
n		input	矩阵 op(A) 和 C 的行数
k		input	矩阵 op(A) 的列数
alpha	主机或设备	input	用于乘法的 <type> 标量
A	设备	input	如果 `trans == MCBLAS_OP_N`，<type> 数组维度为 `lda x k`，其中 `lda>=max(1,n)`；否则，维度为 `lda x n`，其中 `lda>=max(1,k)`
lda		input	用于存储矩阵 A 的二维数组的前导维度
beta	主机或设备	input	用于乘法的 <type> 标量，如果 `beta == 0`，则 C 不必为有效输入
C	设备	in/out	<type> 数组，维度为 `ldc x n`，其中 `ldc>=max(1,n)`
ldc		input	用于存储矩阵 C 的二维数组的前导维度

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_INVALID_VALUE	• n, k < 0 • trans != MCBLAS_OP_N, MCBLAS_OP_C, MCBLAS_OP_T • uplo != MCBLAS_FILL_MODE_LOWER, MCBLAS_FILL_MODE_UPPER • 如果 trans == MCBLAS_OP_N，lda < max(1, n)；否则 lda < max(1, k) • ldc < max(1, n) • alpha == NULL 或 beta == NULL • 如果 C 需要缩放，C = NULL
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.5.7 mcblas<t>syr2k()

```
mcblasStatus_t mcblasSsyr2k(mcblasHandle_t handle,
                            mcblasFillMode_t uplo, mcblasOperation_t trans,
                            int n, int k,
                            const float           *alpha,
                            const float           *A, int lda,
                            const float           *B, int ldb,
                            const float           *beta,
                            float           *C, int ldc)
mcblasStatus_t mcblasDsyr2k(mcblasHandle_t handle,
                            mcblasFillMode_t uplo, mcblasOperation_t trans,
                            int n, int k,
                            const double          *alpha,
                            const double          *A, int lda,
                            const double          *B, int ldb,
                            const double          *beta,
                            double          *C, int ldc)
mcblasStatus_t mcblasCsyr2k(mcblasHandle_t handle,
                            mcblasFillMode_t uplo, mcblasOperation_t trans,
                            int n, int k,
                            const mcComplex       *alpha,
                            const mcComplex       *A, int lda,
                            const mcComplex       *B, int ldb,
                            const mcComplex       *beta,
                            mcComplex       *C, int ldc)
mcblasStatus_t mcblasZsyr2k(mcblasHandle_t handle,
                            mcblasFillMode_t uplo, mcblasOperation_t trans,
                            int n, int k,
                            const mcDoubleComplex *alpha,
                            const mcDoubleComplex *A, int lda,
                            const mcDoubleComplex *B, int ldb,
                            const mcDoubleComplex *beta,
                            mcDoubleComplex *C, int ldc)
```

此函数执行对称秩- $2k$ 校正 $C = \alpha(\text{op}(A)\text{op}(B)^T + \text{op}(B)\text{op}(A)^T) + \beta C$，其中 α 和 β 是标量，C 是以 lower 或 upper 模式存储的对称矩阵，A 和 B 是维度分别为 op(A) $n \times k$ 和 op(B) $n \times k$ 的矩阵。此外，对于矩阵 A 和 B

$$op(A)\ and\ op(B) = \begin{cases} A\ and\ B & if\ trans == MCBLAS\_OP\_N \\ A^T\ and\ B^T & if\ trans == MCBLAS\_OP\_T \end{cases}$$

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
uplo		input	指示存储的是矩阵 C 的上半部分还是下半部分，其对称部分未被引用，且是从存储的元素推断出来的
trans		input	非转置或转置运算 op(A)
n		input	矩阵 op(A)、op(B) 和 C 的行数
k		input	矩阵 op(A) 和 op(B) 的列数
alpha	主机或设备	input	用于乘法的 <type> 标量
A	设备	input	如果 `transa == MCBLAS_OP_N`，<type> 数组维度为 `lda x k`，其中 `lda>=max(1,n)`；否则，维度为 `lda x n`，其中 `lda>=max(1,k)`
lda		input	用于存储矩阵 A 的二维数组的前导维度
B	设备	input	如果 `transb == MCBLAS_OP_N`，<type> 数组维度为 `ldb x k`，其中 `ldb>=max(1,n)`；否则，维度为 `ldb x n`，其中 `ldb>=max(1,k)`

下页继续

表 2.6 – 续上页

参数	内存	In/out	含义
ldb		input	用于存储矩阵 B 的二维数组的前导维度
beta	主机或设备	input	用于乘法的 <type> 标量，如果 beta == 0，则 C 不必为有效输入
C	设备	in/out	<type> 数组，维度为 ldc x n，其中 ldc>=max(1,n)
ldc		input	用于存储矩阵 C 的二维数组的前导维度

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_INVALID_VALUE	• n, k < 0 • trans != MCBLAS_OP_N, MCBLAS_OP_C, MCBLAS_OP_T • uplo != MCBLAS_FILL_MODE_LOWER, MCBLAS_FILL_MODE_UPPER • 如果 trans == MCBLAS_OP_N，lda < max(1, n)；否则 lda < max(1, k) • 如果 trans == MCBLAS_OP_N，ldb < max(1, n)；否则 lda < max(1, k) • ldc < max(1, n) • alpha == NULL 或 beta == NULL • 如果 C 需要缩放，C = NULL
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.5.8 mcblas<t>syrkx()

```
mcblasStatus_t mcblasSsyrkx(mcblasHandle_t handle,
                            mcblasFillMode_t uplo, mcblasOperation_t trans,
                            int n, int k,
                            const float           *alpha,
                            const float           *A, int lda,
                            const float           *B, int ldb,
                            const float           *beta,
                            float           *C, int ldc)
mcblasStatus_t mcblasDsyrkx(mcblasHandle_t handle,
                            mcblasFillMode_t uplo, mcblasOperation_t trans,
                            int n, int k,
                            const double          *alpha,
                            const double          *A, int lda,
                            const double          *B, int ldb,
                            const double          *beta,
                            double          *C, int ldc)
mcblasStatus_t mcblasCsyrkx(mcblasHandle_t handle,
                            mcblasFillMode_t uplo, mcblasOperation_t trans,
                            int n, int k,
                            const mcComplex       *alpha,
                            const mcComplex       *A, int lda,
                            const mcComplex       *B, int ldb,
                            const mcComplex       *beta,
                            mcComplex       *C, int ldc)
```

(下页继续)

(续上页)

```
mcblasStatus_t mcblasZsyrkx(mcblasHandle_t handle,
                            mcblasFillMode_t uplo, mcblasOperation_t trans,
                            int n, int k,
                            const mcDoubleComplex *alpha,
                            const mcDoubleComplex *A, int lda,
                            const mcDoubleComplex *B, int ldb,
                            const mcDoubleComplex *beta,
                            mcDoubleComplex *C, int ldc)
```

此函数执行对称秩-k更新的变体$C = \alpha\text{op}(A)\text{op}(B)^T + \beta C$，其中$\alpha$和$\beta$是标量，$C$是以 lower 或 upper 模式存储的对称矩阵，A和B是维度分别为 op(A) $n \times k$和 op(B) $n \times k$的矩阵。此外，对于矩阵A和B

$$op(A)\ and\ op(B) = \begin{cases} A\ and\ B & if\ trans == MCBLAS\_OP\_N \\ A^T\ and\ B^T & if\ trans == MCBLAS\_OP\_T \end{cases}$$

当矩阵 B 能保证结果是对称矩阵时，可以使用此例程。一个常见的例子是矩阵 B 是矩阵 A 的缩放形式：这相当于 B 是矩阵 A 和一个对角矩阵（diagonal matrix）的乘积。有效计算正则矩阵（regular matrix）与对角矩阵的乘积，请参见mcblas<t>dgmm()。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
uplo		input	指示存储的是矩阵 C 的上半部分还是下半部分，其对称部分未被引用，且是从存储的元素推断出来的
trans		input	非转置或转置运算 op(A)
n		input	矩阵 op(A)、op(B) 和 C 的行数
k		input	矩阵 op(A) 和 op(B) 的列数
alpha	主机或设备	input	用于乘法的 <type> 标量
A	设备	input	如果 `transa == MCBLAS_OP_N`，<type> 数组维度为 `lda x k`，其中 `lda>=max(1,n)`；否则，维度为 `lda x n`，其中 `lda>=max(1,k)`
lda		input	用于存储矩阵 A 的二维数组的前导维度
B	设备	input	如果 `transb == MCBLAS_OP_N`，<type> 数组维度为 `ldb x k`，其中 `ldb>=max(1,n)`；否则，维度为 `ldb x n`，其中 `ldb>=max(1,k)`
ldb		input	用于存储矩阵 B 的二维数组的前导维度
beta	主机或设备	input	用于乘法的 <type> 标量，如果 `beta == 0`，则 C 不必为有效输入
C	设备	in/out	<type> 数组，维度为 `ldc x n`，其中 `ldc>=max(1,n)`
ldc		input	用于存储矩阵 C 的二维数组的前导维度

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_INVALID_VALUE	• n, k < 0 • trans != MCBLAS_OP_N, MCBLAS_OP_C, MCBLAS_OP_T • uplo != MCBLAS_FILL_MODE_LOWER, MCBLAS_FILL_MODE_UPPER • 如果 trans == MCBLAS_OP_N，lda < max(1, n)；否则 lda < max(1, k) • 如果 trans == MCBLAS_OP_N，ldb < max(1, n)；否则 lda < max(1, k) • ldc < max(1, n) • alpha == NULL 或 beta == NULL • 如果 C 需要缩放，C = NULL
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.5.9 mcblas<t>trmm()

```
mcblasStatus_t mcblasStrmm(mcblasHandle_t handle,
                           mcblasSideMode_t side, mcblasFillMode_t uplo,
                           mcblasOperation_t trans, mcblasDiagType_t diag,
                           int m, int n,
                           const float          *alpha,
                           const float          *A, int lda,
                           const float          *B, int ldb,
                           float                *C, int ldc)
mcblasStatus_t mcblasDtrmm(mcblasHandle_t handle,
                           mcblasSideMode_t side, mcblasFillMode_t uplo,
                           mcblasOperation_t trans, mcblasDiagType_t diag,
                           int m, int n,
                           const double         *alpha,
                           const double         *A, int lda,
                           const double         *B, int ldb,
                           double               *C, int ldc)
mcblasStatus_t mcblasCtrmm(mcblasHandle_t handle,
                           mcblasSideMode_t side, mcblasFillMode_t uplo,
                           mcblasOperation_t trans, mcblasDiagType_t diag,
                           int m, int n,
                           const mcComplex      *alpha,
                           const mcComplex      *A, int lda,
                           const mcComplex      *B, int ldb,
                           mcComplex            *C, int ldc)
mcblasStatus_t mcblasZtrmm(mcblasHandle_t handle,
                           mcblasSideMode_t side, mcblasFillMode_t uplo,
                           mcblasOperation_t trans, mcblasDiagType_t diag,
                           int m, int n,
                           const mcDoubleComplex *alpha,
                           const mcDoubleComplex *A, int lda,
                           const mcDoubleComplex *B, int ldb,
                           mcDoubleComplex       *C, int ldc)
```

该函数执行三角矩阵-矩阵乘法

$$C = \begin{cases} \alpha op(A)B & if\ \ side == MCBLAS\_SIDE\_LEFT \\ \alpha Bop(A) & if\ \ side == MCBLAS\_SIDE\_RIGHT \end{cases}$$

其中 A 是以 lower 或 upper 模式存储的三角矩阵（无论是否有主对角线），B 和 C 为 m times n 矩阵，α 是标量。此外，对于矩阵 A，

$$op(A) = \begin{cases} A & if\ \ trans == MCBLAS\_OP\_N \\ A^T & if\ \ trans == MCBLAS\_OP\_T \\ A^H & if\ \ trans == MCBLAS\_OP\_C \end{cases}$$

请注意，为了实现更好的并行性，mcBLAS 与 BLAS API 仅在此例程上不同。BLAS API 采用就地实现（结果写回到 B），而 mcBLAS API 采用异地实现（结果写入 C）。通过传递矩阵 B 的地址来代替矩阵 C，此应用可以在 mcBLAS API 中获得 BLAS 的就地实现功能。不支持其他输入参数的重叠。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
side		input	指示矩阵 A 是在 B 的左边还是右边
uplo		input	指示存储的是矩阵 A 的上半部分还是下半部分，另外半部分未被引用，且是从存储的元素推断出来的
trans		input	非转置或共轭转置运算 op(A)
diag		input	指示矩阵 A 主对角线上的元素是否是统一的且不应被访问
m		input	矩阵 B 的行数，矩阵 A 的大小也相应调整
n		input	矩阵 B 的列数，矩阵 A 的大小也相应调整
alpha	主机或设备	input	用于乘法的 <type> 标量，如果 alpha == 0，则 A 未被引用，且 B 不必为有效输入
A	设备	input	如果 side == MCBLAS_SIDE_LEFT ，<type> 数组维度为 lda x m，其中 lda>=max(1,m)；否则，维度为 lda x n，其中 lda>=max(1,n)
lda		input	用于存储矩阵 A 的二维数组的前导维度
B	设备	input	<type> 数组，维度为 ldb x n，其中 ldb>=max(1,m)
ldb		input	用于存储矩阵 B 的二维数组的前导维度
C	设备	in/out	<type> 数组，维度为 ldc x n，其中 ldc>=max(1,m)
ldc		input	用于存储矩阵 C 的二维数组的前导维度

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_INVALID_VALUE	• m, n < 0 • trans != MCBLAS_OP_N, MCBLAS_OP_C, MCBLAS_OP_T • uplo != MCBLAS_FILL_MODE_LOWER, MCBLAS_FILL_MODE_UPPER • side != MCBLAS_SIDE_LEFT, MCBLAS_SIDE_RIGHT • 如果 side == MCBLAS_SIDE_LEFT ，lda < max(1, m)；否则 lda < max(1, n) • ldb < max(1, m) • alpha == NULL 或 beta == NULL • 如果 C 需要缩放，C = NULL
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.5.10 mcblas<t>trsm()

```
mcblasStatus_t mcblasStrsm(mcblasHandle_t handle,
                           mcblasSideMode_t side, mcblasFillMode_t uplo,
                           mcblasOperation_t trans, mcblasDiagType_t diag,
                           int m, int n,
                           const float           *alpha,
                           const float           *A, int lda,
                           float                 *B, int ldb)
mcblasStatus_t mcblasDtrsm(mcblasHandle_t handle,
                           mcblasSideMode_t side, mcblasFillMode_t uplo,
                           mcblasOperation_t trans, mcblasDiagType_t diag,
                           int m, int n,
                           const double          *alpha,
                           const double          *A, int lda,
                           double                *B, int ldb)
mcblasStatus_t mcblasCtrsm(mcblasHandle_t handle,
                           mcblasSideMode_t side, mcblasFillMode_t uplo,
                           mcblasOperation_t trans, mcblasDiagType_t diag,
                           int m, int n,
                           const mcComplex       *alpha,
                           const mcComplex       *A, int lda,
                           mcComplex             *B, int ldb)
mcblasStatus_t mcblasZtrsm(mcblasHandle_t handle,
                           mcblasSideMode_t side, mcblasFillMode_t uplo,
                           mcblasOperation_t trans, mcblasDiagType_t diag,
                           int m, int n,
                           const mcDoubleComplex *alpha,
                           const mcDoubleComplex *A, int lda,
                           mcDoubleComplex       *B, int ldb)
```

该函数使用以下公式求解三角线性方程组

$$\begin{cases} op(A)X = \alpha B & if\ \ side == MCBLAS\_SIDE\_LEFT \\ Xop(A) = \alpha B & if\ \ side == MCBLAS\_SIDE\_RIGHT \end{cases}$$

其中 A 是以 lower 或 upper 模式存储的三角矩阵（无论是否有主对角线），X 和 B 为 m times n 矩阵，α 是标量。此外，对于矩阵 A，

$$op(A) = \begin{cases} A & if\ \ trans == MCBLAS\_OP\_N \\ A^T & if\ \ trans == MCBLAS\_OP\_T \\ A^H & if\ \ trans == MCBLAS\_OP\_C \end{cases}$$

结束时 X 覆盖右侧 B。

此函数中不包括奇异性或接近奇异性的测试。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
side		input	指示矩阵 A 是在 X 的左边还是右边
uplo		input	指示存储的是矩阵 A 的上半部分还是下半部分，另外半部分未被引用，且是从存储的元素推断出来的
trans		input	非转置或共轭转置运算 op(A)
diag		input	指示矩阵 A 主对角线上的元素是否是统一的且不应被访问
m		input	矩阵 B 的行数，矩阵 A 的大小也相应调整
n		input	矩阵 B 的列数，矩阵 A 的大小也相应调整
alpha	主机或设备	input	用于乘法的 <type> 标量，如果 alpha == 0，则 A 未被引用，且 B 不必为有效输入
A	设备	input	如果 side == MCBLAS_SIDE_LEFT ，<type> 数组维度为 lda x m，其中 lda>=max(1,m)；否则，维度为 lda x n，其中 lda>=max(1,n)
lda		input	用于存储矩阵 A 的二维数组的前导维度
B	设备	in/out	<type> 数组，维度为 ldb x n，其中 ldb>=max(1,m)
ldb		input	用于存储矩阵 B 的二维数组的前导维度

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_INVALID_VALUE	• m < 0 或 n < 0 • trans != MCBLAS_OP_N, MCBLAS_OP_C, MCBLAS_OP_T • uplo != MCBLAS_FILL_MODE_LOWER, MCBLAS_FILL_MODE_UPPER • side != MCBLAS_SIDE_LEFT, MCBLAS_SIDE_RIGHT • diag != MCBLAS_DIAG_NON_UNIT, MCBLAS_DIAG_UNIT • 如果 side == MCBLAS_SIDE_LEFT ，lda < max(1, m)；否则 lda < max(1, n) • ldb < max(1, m) • alpha == NULL
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.5.11 mcblas<t>trsmBatched()

```
mcblasStatus_t mcblasStrsmBatched(mcblasHandle_t    handle,
                                  mcblasSideMode_t  side,
                                  mcblasFillMode_t  uplo,
                                  mcblasOperation_t trans,
                                  mcblasDiagType_t  diag,
                                  int m,
                                  int n,
                                  const float *alpha,
                                  const float *const A[],
                                  int lda,
                                  float *const B[],
                                  int ldb,
```

(下页继续)

(续上页)

```
                                    int batchCount);
mcblasStatus_t mcblasDtrsmBatched(mcblasHandle_t    handle,
                                    mcblasSideMode_t  side,
                                    mcblasFillMode_t  uplo,
                                    mcblasOperation_t trans,
                                    mcblasDiagType_t  diag,
                                    int m,
                                    int n,
                                    const double *alpha,
                                    const double *const A[],
                                    int lda,
                                    double *const B[],
                                    int ldb,
                                    int batchCount);
mcblasStatus_t mcblasCtrsmBatched(mcblasHandle_t    handle,
                                    mcblasSideMode_t  side,
                                    mcblasFillMode_t  uplo,
                                    mcblasOperation_t trans,
                                    mcblasDiagType_t  diag,
                                    int m,
                                    int n,
                                    const mcComplex *alpha,
                                    const mcComplex *const A[],
                                    int lda,
                                    mcComplex *const B[],
                                    int ldb,
                                    int batchCount);
mcblasStatus_t mcblasZtrsmBatched(mcblasHandle_t    handle,
                                    mcblasSideMode_t  side,
                                    mcblasFillMode_t  uplo,
                                    mcblasOperation_t trans,
                                    mcblasDiagType_t  diag,
                                    int m,
                                    int n,
                                    const mcDoubleComplex *alpha,
                                    const mcDoubleComplex *const A[],
                                    int lda,
                                    mcDoubleComplex *const B[],
                                    int ldb,
                                    int batchCount);
```

该函数使用以下公式求解三角线性方程组数组

$$\begin{cases} op(A[i])X[i] = \alpha B[i] & if\ \ side == MCBLAS\_SIDE\_LEFT \\ X[i]op(A[i]) = \alpha B[i] & if\ \ side == MCBLAS\_SIDE\_RIGHT \end{cases}$$

其中 $A[i]$ 是以 lower 或 upper 模式存储的三角矩阵（无论是否有主对角线），$X[i]$ 和 $B[i]$ 为 m times n 矩阵，α 是标量。此外，对于矩阵 A

$$op(A[i]) = \begin{cases} A[i] & if\ \ transa == MCBLAS\_OP\_N \\ A^T[i] & if\ \ transa == MCBLAS\_OP\_T \\ A^H[i] & if\ \ transa == MCBLAS\_OP\_C \end{cases}$$

结束时 $X[i]$ 覆盖右侧 $B[i]$。此函数中不包括奇异性或接近奇异性的测试。

此函数适用于任何大小，但主要用于较小的矩阵，其中启动开销（launch overhead）是一个重要因素。对于较大的矩阵，可能需要在一组 MXMACA 流中调用 `batchCount` 乘以正则 `mcblas<t>trsm`。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
side		input	指示矩阵 A[i] 在 X[i] 的左边还是右边
uplo		input	指示存储的是矩阵 A[i] 的上半部分还是下半部分，另外半部分未被引用，且是从存储的元素推断出来的
trans		input	非转置或共轭转置运算 op(A[i])
diag		input	指示矩阵 A[i] 主对角线上的元素是否是统一的且不应被访问
m		input	矩阵 B[i] 的行数，矩阵 A[i] 的大小也相应调整
n		input	矩阵 B[i] 的列数，矩阵 A[i] 的大小也相应调整
alpha	主机或设备	input	用于乘法的 <type> 标量，如果 alpha == 0，则 A[i] 未被引用，且 B[i] 不必为有效输入
A	设备	input	指向 <type> 数组的指针数组。如果 side == MCBLAS_SIDE_LEFT，数组维度为 lda x m，其中 lda>=max(1,m)；否则，维度为 lda x n，其中 lda>=max(1,n)
lda		input	用于存储矩阵 A[i] 的二维数组的前导维度
B	设备	in/out	指向 <type> 数组的指针数组，维度为 ldb x n，其中 ldb>=max(1,m)。矩阵 B[i] 不应重叠；否则，将出现未定义的行为
ldb		input	用于存储矩阵 B[i] 的二维数组的前导维度
batchCount		input	A 和 B 中包含的指针数

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_INVALID_VALUE	• m < 0,n < 0 或 batchCount < 0 • trans != MCBLAS_OP_N, MCBLAS_OP_C, MCBLAS_OP_T • uplo != MCBLAS_FILL_MODE_LOWER, MCBLAS_FILL_MODE_UPPER • side != MCBLAS_SIDE_LEFT, MCBLAS_SIDE_RIGHT • diag != MCBLAS_DIAG_NON_UNIT, MCBLAS_DIAG_UNIT • 如果 side == MCBLAS_SIDE_LEFT ，lda < max(1, m)；否则 lda < max(1, n) • ldb < max(1, m)
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.5.12 mcblas<t>hemm()

```
mcblasStatus_t mcblasChemm(mcblasHandle_t handle,
                           mcblasSideMode_t side, mcblasFillMode_t uplo,
                           int m, int n,
                           const mcComplex       *alpha,
                           const mcComplex       *A, int lda,
                           const mcComplex       *B, int ldb,
                           const mcComplex       *beta,
                           mcComplex             *C, int ldc)
```

(下页继续)

(续上页)

```
mcblasStatus_t mcblasZhemm(mcblasHandle_t handle,
                           mcblasSideMode_t side, mcblasFillMode_t uplo,
                           int m, int n,
                           const mcDoubleComplex *alpha,
                           const mcDoubleComplex *A, int lda,
                           const mcDoubleComplex *B, int ldb,
                           const mcDoubleComplex *beta,
                           mcDoubleComplex       *C, int ldc)
```

该函数执行厄米矩阵-矩阵乘法

$$C = \begin{cases} \alpha AB + \beta C & if\ \ side == MCBLAS\_SIDE\_LEFT \\ \alpha BA + \beta C & if\ \ side == MCBLAS\_SIDE\_RIGHT \end{cases}$$

其中 A 是以 lower 或 upper 模式存储的厄米矩阵，B 和 C 是 $m \times n$ 矩阵，α 和 β 是标量。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
side		input	指示矩阵 A 在 B 的左边还是右边
uplo		input	指示存储的是矩阵 A 的上半部分还是下半部分，另外 Hermitian 部分未被引用，且是从存储的元素推断出来的
m		input	矩阵 C 和 B 的行数，矩阵 A 的大小也相应调整
n		input	矩阵 C 和 B 的列数，矩阵 A 的大小也相应调整
alpha	主机或设备	input	用于乘法的 <type> 标量，
A	设备	input	如果 `side == MCBLAS_SIDE_LEFT` ，<type> 数组维度为 `lda x m` ，其中 `lda>=max(1,m)` ；否则，维度为 `lda x n` ，其中 `lda>=max(1,n)` 假定对角线元素的虚数部分为 0
lda		input	用于存储矩阵 A 的二维数组的前导维度
B	设备	input	<type> 数组，维度为 `ldb x n` 其中 `ldb>=max(1,m)`
ldb		input	用于存储矩阵 B 的二维数组的前导维度
beta		input	用于乘法的 <type> 标量，如果 `beta == 0` ，则 C 不必为有效输入
C	设备	in/out	<type> 数组，维度为 `ldc x n` 其中 `ldc>=max(1,m)`
ldc		input	用于存储矩阵 C 的二维数组的前导维度

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_INVALID_VALUE	• m < 0 或 n < 0 • side != MCBLAS_SIDE_LEFT, MCBLAS_SIDE_RIGHT • uplo != MCBLAS_FILL_MODE_LOWER, MCBLAS_FILL_MODE_UPPER • 如果 side == MCBLAS_SIDE_LEFT ，lda < max(1, m)；否则 lda < max(1, n) • ldb < max(1, m) • ldc < max(1, m) • alpha == NULL 或 beta == NULL • C = NULL
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.5.13 mcblas<t>herk()

```
mcblasStatus_t mcblasCherk(mcblasHandle_t handle,
                           mcblasFillMode_t uplo, mcblasOperation_t trans,
                           int n, int k,
                           const float  *alpha,
                           const mcComplex       *A, int lda,
                           const float  *beta,
                           mcComplex       *C, int ldc)
mcblasStatus_t mcblasZherk(mcblasHandle_t handle,
                           mcblasFillMode_t uplo, mcblasOperation_t trans,
                           int n, int k,
                           const double *alpha,
                           const mcDoubleComplex *A, int lda,
                           const double *beta,
                           mcDoubleComplex *C, int ldc)
```

此函数执行厄米秩-k 校正 $C = \alpha \text{op}(A)\text{op}(A)^H + \beta C$，其中 α 和 β 是标量，C 是以 lower 或 upper 模式存储的厄米矩阵，A 是一个维度为 op(A) $n \times k$ 的矩阵。此外，对于矩阵 A

$$op(A) = \begin{cases} A & if\ transa == MCBLAS\_OP\_N \\ A^T & if\ transa == MCBLAS\_OP\_T \end{cases}$$

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
uplo		input	指示存储的是矩阵 A 的上半部分还是下半部分，另外 Hermitian 部分未被引用，且是从存储的元素推断出来的
trans		input	非转置或共轭转置运算 op(A)
n		input	矩阵 op(A) 和 C 的行数
alpha	主机或设备	input	用于乘法的 <type> 标量
A	设备	input	如果 `transa == MCBLAS_OP_N`，<type> 数组维度为 `lda x k`，其中 `lda>=max(1,n)`；否则，维度为 `lda x n`，其中 `lda>=max(1,k)`
lda		input	用于存储矩阵 A 的二维数组的前导维度
beta		input	用于乘法的 <type> 标量，如果 `beta == 0`，则 C 不必为有效输入
C	设备	in/out	<type> 数组，维度为 `ldc x n` 其中 `ldc>=max(1,n)`
ldc		input	用于存储矩阵 C 的二维数组的前导维度

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_INVALID_VALUE	• n < 0 或 k < 0 • trans != MCBLAS_OP_N, MCBLAS_OP_C, MCBLAS_OP_T • uplo != MCBLAS_FILL_MODE_LOWER, MCBLAS_FILL_MODE_UPPER • 如果 trans == MCBLAS_OP_N，lda < max(1, n)；否则 lda < max(1, k) • ldc < max(1, n) • alpha == NULL 或 beta == NULL • C = NULL
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.5.14 mcblas<t>her2k()

```
mcblasStatus_t mcblasCher2k(mcblasHandle_t handle,
                            mcblasFillMode_t uplo, mcblasOperation_t trans,
                            int n, int k,
                            const mcComplex       *alpha,
                            const mcComplex       *A, int lda,
                            const mcComplex       *B, int ldb,
                            const float  *beta,
                            mcComplex       *C, int ldc)
mcblasStatus_t mcblasZher2k(mcblasHandle_t handle,
                            mcblasFillMode_t uplo, mcblasOperation_t trans,
                            int n, int k,
                            const mcDoubleComplex *alpha,
                            const mcDoubleComplex *A, int lda,
                            const mcDoubleComplex *B, int ldb,
                            const double *beta,
                            mcDoubleComplex *C, int ldc)
```

此函数执行厄米秩- $2k$ 校正 $C = \alpha \text{op}(A)\text{op}(B)^H + \overline{\alpha}\text{op}(B)\text{op}(A)^H + \beta C$，其中 α 和 β 是标量，C 是以 lower 或 upper 模式存储的厄米矩阵，A 和 B 是维度分别为 op(A) $n \times k$ 和 op(B) $n \times k$ 的矩阵。此外，对于矩阵 A 和 B

$$op(A)\ and\ op(B) = \begin{cases} A\ and\ B & if\ trans == MCBLAS\_OP\_N \\ A^H\ and\ B^H & if\ trans == MCBLAS\_OP\_C \end{cases}$$

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
uplo		input	指示存储的是矩阵 A 的上半部分还是下半部分，另外 Hermitian 部分未被引用，且是从存储的元素推断出来的
trans		input	非转置或共轭转置运算 op(A)
n		input	矩阵 op(A)、op(B) 和 C 的行数
k		input	矩阵 op(A) 和 op(B) 的列数
alpha	主机或设备	input	用于乘法的 <type> 标量
A	设备	input	如果 transa == MCBLAS_OP_N，<type> 数组维度为 lda x k，其中 lda>=max(1,n)；否则，维度为 lda x n，其中 lda>=max(1,k)
lda		input	用于存储矩阵 A 的二维数组的前导维度
B	设备	input	如果 transb == MCBLAS_OP_N，<type> 数组维度为 ldb x k，其中 ldb>=max(1,n)；否则，维度为 ldb x n，其中 ldb>=max(1,k)
ldb		input	用于存储矩阵 B 的二维数组的前导维度
beta		input	用于乘法的 <type> 标量，如果 beta == 0，则 C 不必为有效输入
C	设备	in/out	<type> 数组，维度为 ldc x n 其中 ldc>=max(1,n)。假定对角线元素的虚数部分设置为 0
ldc		input	用于存储矩阵 C 的二维数组的前导维度

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_INVALID_VALUE	• n < 0 或 k < 0 • trans != MCBLAS_OP_N, MCBLAS_OP_C, MCBLAS_OP_T • uplo != MCBLAS_FILL_MODE_LOWER, MCBLAS_FILL_MODE_UPPER • 如果 trans == MCBLAS_OP_N，lda < max(1, n)；否则 lda < max(1, k) • 如果 trans == MCBLAS_OP_N，ldb < max(1, n)；否则 lda < max(1, k) • ldc < max(1, n) • alpha == NULL 或 beta == NULL • C = NULL
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.5.15 mcblas<t>herkx()

```
mcblasStatus_t mcblasCherkx(mcblasHandle_t handle,
                            mcblasFillMode_t uplo, mcblasOperation_t trans,
                            int n, int k,
                            const mcComplex       *alpha,
                            const mcComplex       *A, int lda,
                            const mcComplex       *B, int ldb,
                            const float  *beta,
                            mcComplex       *C, int ldc)
mcblasStatus_t mcblasZherkx(mcblasHandle_t handle,
                            mcblasFillMode_t uplo, mcblasOperation_t trans,
                            int n, int k,
                            const mcDoubleComplex *alpha,
                            const mcDoubleComplex *A, int lda,
                            const mcDoubleComplex *B, int ldb,
                            const double *beta,
                            mcDoubleComplex *C, int ldc)
```

此函数执行厄米秩-k 校正变分 $C = \alpha \text{op}(A) \text{op}(B)^H + \beta C$，其中 α 和 β 是标量，C 是以 lower 或 upper 模式存储的厄米矩阵，A 和 B 是维度分别为 op(A) $n \times k$ 和 op(B) $n \times k$ 的矩阵。此外，对于矩阵 A 和 B

$$op(A) \ and \ op(B) = \begin{cases} A \ and \ B & if \ trans == MCBLAS\_OP\_N \\ A^H \ and \ B^H & if \ trans == MCBLAS\_OP\_C \end{cases}$$

当矩阵 B 的结果保证是厄米矩阵，可以使用此例程。一个常见的例子是矩阵 B 是矩阵 A 的缩放形式：这相当于 B 是矩阵 A 和一个对角矩阵（diagonal matrix）的乘积。有效计算正则矩阵（regular matrix）与对角矩阵的乘积，请参见mcblas<t>dgmm()。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
uplo		input	指示存储的是矩阵 A 的上半部分还是下半部分，另外 Hermitian 部分未被引用，且是从存储的元素推断出来的
trans		input	非转置或共轭转置运算 op(A)
n		input	矩阵 op(A)、op(B) 和 C 的行数
k		input	矩阵 op(A) 和 op(B) 的列数
alpha	主机或设备	input	用于乘法的 <type> 标量
A	设备	input	如果 transa == MCBLAS_OP_N，<type> 数组维度为 lda x k，其中 lda>=max(1,n)；否则，维度为 lda x n，其中 lda>=max(1,k)
lda		input	用于存储矩阵 A 的二维数组的前导维度
B	设备	input	如果 transb == MCBLAS_OP_N，<type> 数组维度为 ldb x k，其中 ldb>=max(1,n)；否则，维度为 ldb x n，其中 ldb>=max(1,k)
ldb		input	用于存储矩阵 B 的二维数组的前导维度
beta	主机或设备	input	用于乘法的 <type> 标量，如果 beta == 0，则 C 不必为有效输入
C	设备	in/out	<type> 数组，维度为 ldc x n 其中 ldc>=max(1,n)。假定对角线元素的虚数部分设置为 0
ldc		input	用于存储矩阵 C 的二维数组的前导维度

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_INVALID_VALUE	• n < 0 或 k < 0 • trans != MCBLAS_OP_N, MCBLAS_OP_C, MCBLAS_OP_T • uplo != MCBLAS_FILL_MODE_LOWER, MCBLAS_FILL_MODE_UPPER • 如果 trans == MCBLAS_OP_N，lda < max(1, n)；否则 lda < max(1, k) • 如果 trans == MCBLAS_OP_N，ldb < max(1, n)；否则 lda < max(1, k) • ldc < max(1, n) • alpha == NULL 或 beta == NULL • C = NULL
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.6 BLAS-Like 扩展函数

本章中，我们将介绍执行矩阵-矩阵操作的 BLAS 扩展函数。

2.6.1 mcblas<t>geam()

```
mcblasStatus_t mcblasSgeam(mcblasHandle_t handle,
                           mcblasOperation_t transa, mcblasOperation_t
↪transb,
```

(下页继续)

(续上页)

```
                            int m, int n,
                            const float           *alpha,
                            const float           *A, int lda,
                            const float           *beta,
                            const float           *B, int ldb,
                            float           *C, int ldc)
mcblasStatus_t mcblasDgeam(mcblasHandle_t handle,
                            mcblasOperation_t transa, mcblasOperation_t
 ↪transb,
                            int m, int n,
                            const double          *alpha,
                            const double          *A, int lda,
                            const double          *beta,
                            const double          *B, int ldb,
                            double          *C, int ldc)
mcblasStatus_t mcblasCgeam(mcblasHandle_t handle,
                            mcblasOperation_t transa, mcblasOperation_t
 ↪transb,
                            int m, int n,
                            const mcComplex       *alpha,
                            const mcComplex       *A, int lda,
                            const mcComplex       *beta ,
                            const mcComplex       *B, int ldb,
                            mcComplex       *C, int ldc)
mcblasStatus_t mcblasZgeam(mcblasHandle_t handle,
                            mcblasOperation_t transa, mcblasOperation_t
 ↪transb,
                            int m, int n,
                            const mcDoubleComplex *alpha,
                            const mcDoubleComplex *A, int lda,
                            const mcDoubleComplex *beta,
                            const mcDoubleComplex *B, int ldb,
                            mcDoubleComplex *C, int ldc)
```

该函数执行矩阵-矩阵加法/转置

$$C = \alpha \mathrm{op}(A) + \beta \mathrm{op}(B)$$

其中 α 和 β 是标量，A、B、C 是以列主格式存储的矩阵，维度分别为 op(A) $m \times n$、op(B) $m \times n$ 和 C $m \times n$。此外，对于矩阵 A

$$op(A) = \begin{cases} A & if\ transa == MCBLAS\_OP\_N \\ A^T & if\ transa == MCBLAS\_OP\_T \\ A^H & if\ transa == MCBLAS\_OP\_C \end{cases}$$

对于矩阵 B，op(B) 的定义类似。如果 C 与 A 或 B 不重叠，则该操作为异地操作。

就地模式支持以下两种操作：

$$C = \alpha * C + \beta \mathrm{op}(B)$$
$$C = \alpha \mathrm{op}(A) + \beta * C$$

对于就地模式，如果 `C = A`，则 `ldc = lda`，`transa = MCBLAS_OP_N`。如果 `C = B`，则 `ldc = ldb`，`transb = MCBLAS_OP_N`。如果不满足上述要求，则返回 `MCBLAS_STATUS_INVALID_VALUE`。

该操作包括以下特殊情况：

用户可以通过设置 `*alpha=*beta=0` 将矩阵 C 重置为零。

用户可以通过设置 `*alpha=1 and *beta=0` 对矩阵 A 进行转置。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
transa		input	非转置或共轭转置运算 op(A)
transb		input	非转置或共轭转置运算 op(B)
m		input	矩阵 op(A) 和 C 的行数
n		input	矩阵 op(B) 和 C 的列数
alpha	主机或设备	input	用于乘法的 <type> 标量。如果 *alpha == 0，A 不必为有效输入
A	设备	input	如果 transa == MCBLAS_OP_N，<type> 数组维度为 lda x n，其中 lda>=max(1,m)；否则，维度为 lda x m，其中 lda>=max(1,n)
lda		input	用于存储矩阵 A 的二维数组的前导维度
B	设备	input	如果 transb == MCBLAS_OP_N，<type> 数组维度为 ldb x n，其中 ldb>=max(1,m)；否则，维度为 ldb x m，其中 ldb>=max(1,n)
ldb		input	用于存储矩阵 B 的二维数组的前导维度
beta	主机或设备	input	用于乘法的 <type> 标量，如果 *beta == 0，则 B 不必为有效输入
C	设备	output	<type> 数组，维度为 ldc x n 其中 ldc>=max(1,m)
ldc		input	用于存储矩阵 C 的二维数组的前导维度

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_INVALID_VALUE	• m < 0 或 n < 0 • transa != MCBLAS_OP_N, MCBLAS_OP_C, MCBLAS_OP_T • transb != MCBLAS_OP_N, MCBLAS_OP_C, MCBLAS_OP_T • 如果 transa == MCBLAS_OP_N，lda < max(1, m)；否则 lda < max(1, n) • 如果 transb == MCBLAS_OP_N，ldb < max(1, m)；否则 ldb < max(1, n) • ldc < max(1, m) • A == C, ((MCBLAS_OP_N != transa) || (lda != ldc)) • B == C, ((MCBLAS_OP_N != transb) || (ldb != ldc)) • alpha == NULL 或 beta == NULL
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.6.2 mcblas<t>dgmm()

```
mcblasStatus_t mcblasSdgmm(mcblasHandle_t handle, mcblasSideMode_t mode,
                           int m, int n,
                           const float           *A, int lda,
                           const float           *x, int incx,
                           float           *C, int ldc)
mcblasStatus_t mcblasDdgmm(mcblasHandle_t handle, mcblasSideMode_t mode,
                           int m, int n,
                           const double          *A, int lda,
                           const double          *x, int incx,
```

(下页继续)

(续上页)

```
                            double          *C, int ldc)
mcblasStatus_t mcblasCdgmm(mcblasHandle_t handle, mcblasSideMode_t mode,
                            int m, int n,
                            const mcComplex       *A, int lda,
                            const mcComplex       *x, int incx,
                            mcComplex       *C, int ldc)
mcblasStatus_t mcblasZdgmm(mcblasHandle_t handle, mcblasSideMode_t mode,
                            int m, int n,
                            const mcDoubleComplex *A, int lda,
                            const mcDoubleComplex *x, int incx,
                            mcDoubleComplex *C, int ldc)
```

该函数执行矩阵-矩阵乘法

$$C = \begin{cases} A \times diag(X) & if\ \ mode == MCBLAS\_SIDE\_RIGHT \\ diag(X) \times A & if\ \ mode == MCBLAS\_SIDE\_LEFT \end{cases}$$

其中 A 和 C 是以列主格式存储的矩阵，维度为 $m \times n$。若 `mode == MCBLAS_SIDE_RIGHT`，则 X 是大小为 n 的向量；若 `mode == MCBLAS_SIDE_LEFT`，则是大小为 m 的向量。X 是从步幅（stride）为 `incx` 的一维数组 x 中获取的。`incx` 的绝对值为步幅，`incx` 的符号是步幅的方向。如果 `incx` 是正的，则从第一个元素开始正向读取数组 x。否则，从最后一个元素开始反向读取数组 x。X 的公式为

$$X[j] = \begin{cases} x[j \times incx] & \text{if } incx \geq 0 \\ x[(\chi - 1) \times |incx| - j \times |incx|] & \text{if } incx < 0 \end{cases}$$

其中，如果 `mode == MCBLAS_SIDE_LEFT`，则 $\chi = m$；如果 `mode == MCBLAS_SIDE_RIGHT`，则 $\chi = n$。示例 1：如果用户要执行 $diag(diag(B)) \times A$，则 $incx = ldb + 1$，其中 ldb 是矩阵 `B` 的前导维度，可以是行主格式或列主格式。示例 2：如果用户要执行 $\alpha \times A$，则有两个选项，mcblasgeam（`*beta=0`，`transa == MCBLAS_OP_N`）或 mcblasdgmm（`incx=0`，`x[0]=alpha`）。该操作为异地操作。仅当 `lda = ldc` 时，可使用就地操作。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
mode		input	如果 `mode ==MCBLAS_SIDE_left` ，则左乘；如果 `mode ==MCBLAS_SID E_right` ，则右乘
m		input	矩阵 A 和 C 的行数
n		input	矩阵 A 和 C 的列数
A	设备	in/out	<type> 数组，维度为 `lda x n` 其中 `lda>=max(1,m)`
lda		input	用于存储矩阵 `A` 的二维数组的前导维度
x	设备	input	如果 `mode == MCBLAS_SIDE_LEFT` ，一维 <type> 数组大小为 $\|inc\| \times m$ ；如果 `mode == MCBLAS_SIDE_RIGHT`，大小为 $\|inc\| \times n$
incx		input	每个一维数组 `x` 的步幅
C	设备	in/out	<type> 数组，维度为 `ldc x n` 其中 `ldc>=max(1,m)`
ldc		input	用于存储矩阵 `C` 的二维数组的前导维度

下面列出了此函数可能的返回值及其含义。

返回值	含义
`MCBLAS_STATUS_SUCCESS`	操作成功
`MCBLAS_STATUS_NOT_INITIALIZED`	库未初始化
`MCBLAS_STATUS_INVALID_VALUE`	• `m < 0` 或 `n < 0` • `mode != MCBLAS_SIDE_LEFT, MCBLAS_SIDE_RIGHT` • `lda < max(1, m)` 或 ‘ ‘`ldc < max(1, m)`
`MCBLAS_STATUS_EXECUTION_FAILED`	此函数在 GPU 上启用失败

2.6.3 mcblas<t>getrfBatched()

```
mcblasStatus_t mcblasSgetrfBatched(mcblasHandle_t handle,
                                   int n,
                                   float *const Aarray[],
                                   int lda,
                                   int *PivotArray,
                                   int *infoArray,
                                   int batchSize);

mcblasStatus_t mcblasDgetrfBatched(mcblasHandle_t handle,
                                   int n,
                                   double *const Aarray[],
                                   int lda,
                                   int *PivotArray,
                                   int *infoArray,
                                   int batchSize);

mcblasStatus_t mcblasCgetrfBatched(mcblasHandle_t handle,
                                   int n,
                                   mcComplex *const Aarray[],
                                   int lda,
                                   int *PivotArray,
                                   int *infoArray,
                                   int batchSize);

mcblasStatus_t mcblasZgetrfBatched(mcblasHandle_t handle,
                                   int n,
                                   mcDoubleComplex *const Aarray[],
                                   int lda,
                                   int *PivotArray,
                                   int *infoArray,
                                   int batchSize);
```

`Aarray` 是一个指向矩阵的指针数组，以列主格式存储，维度为 `nxn`，前导维度为 `lda`。

通过以下公式，该函数对每个 `Aarray[i]` 执行 LU 分解，i = 0，…，`batchSize-1`

$$P^{*}Aarray[i] = L^{*}U$$

其中 `P` 是一个置换矩阵，表示进行行交换的部分主元法（partial pivoting）。`L` 是具有单位对角元素的下三角矩阵，`U` 是上三角矩阵。

`P` 为置换矩阵 `Pj` 的积，`j = 1,2,...,n`，即 `P = P1 * P2 * P3 * .... * Pn`。`Pj` 为置换矩阵，在执行 `Pj*x` 时交换向量 x 的两行。通过以下 matlab 代码，`Pj` 可由 `PivotArray[i]` 的 `j` 元素构造

```
//在 Matlab 中，PivotArray[i] 为 base-1 数组。
//在 C 中，PivotArray[i] 为 base-0 数组。
Pj = eye(n);
swap Pj(j,:) and Pj(PivotArray[i][j]  ,:)
```

将 `L` 和 `U` 写回原始矩阵 `A`，丢弃 `L` 的对角元素。`L` 和 `U` 可以通过以下 matlab 代码构造：

```
// A 是 getrf 后的 nxn 矩阵。
L = eye(n);
for j = 1:n
      L(j+1:n,j) = A(j+1:n,j)
end
U = zeros(n);
```

(下页继续)

(续上页)

```
for i = 1:n
      U(i,i:n) = A(i,i:n)
end
```

如果 A(=Aarray[i]) 为奇异矩阵，仍然可使用 getrf，并且 info(=infoArray[i]) 的值表示无法进行 LU 分解的第一行索引。如果 info 为 k，则 U(k,k) 为零。方程 P*A=L*U 仍然有效，但需要不同的 matlab 代码进行 L 和 U 重建，如下所示：

```
// A 是 getrf 后的 nxn 矩阵。
// info 为 k，即 U(k,k) 为零。
L = eye(n);
for j = 1:k-1
      L(j+1:n,j) = A(j+1:n,j)
end
U = zeros(n);
for i = 1:k-1
      U(i,i:n) = A(i,i:n)
end
for i = k:n
      U(i,k:n) = A(i,k:n)
end
```

此函数用于较小的矩阵，其中启动开销是一个重要因素。

如果 PivotArray 为 nil，则 mcblas<t>getrfBatched 支持非主元（non-pivot）LU 分解。

mcblas<t>getrfBatched 支持任意维度。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
n		input	Aarray[i] 的行数和列数
Aarray	设备	in/out	指向 <type> 数组的指针数组，每个数组维度为 n x n，其中 lda >=max(1,n) 。矩阵 Aarray[i] 不应重叠；否则，将出现未定义的行为。
lda		input	用于存储矩阵 Aarray[i] 的二维数组的前导维度
PivotArray	设备	output	大小为 n x batchSize 的数组，包含以线性方式存储的 Aarray[i] 每个因式分解的旋转序列。如果 PivotArray 为零，则禁用旋转
infoArray	设备	output	大小为 batchSize 的数组，info(=infoArray[i]) 包含 Aarray[i] 的因式分解信息。如果 info=0，执行成功。 如果 info=0，执行成功。 如果 info = -j，第 j 个参数具有非法值。 如果 info = k，U(k,k) 为 0。因子分解已经完成，但 U 恰好是奇异的。
batchSize		input	A 中包含的指针数

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_INVALID_VALUE	参数 n,batchSize,lda <0
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.6.4 mcblas<t>getrsBatched()

```
mcblasStatus_t mcblasSgetrsBatched(mcblasHandle_t handle,
                                   mcblasOperation_t trans,
                                   int n,
                                   int nrhs,
                                   const float *const Aarray[],
                                   int lda,
                                   const int *devIpiv,
                                   float *const Barray[],
                                   int ldb,
                                   int *info,
                                   int batchSize);

mcblasStatus_t mcblasDgetrsBatched(mcblasHandle_t handle,
                                   mcblasOperation_t trans,
                                   int n,
                                   int nrhs,
                                   const double *const Aarray[],
                                   int lda,
                                   const int *devIpiv,
                                   double *const Barray[],
                                   int ldb,
                                   int *info,
                                   int batchSize);

mcblasStatus_t mcblasCgetrsBatched(mcblasHandle_t handle,
                                   mcblasOperation_t trans,
                                   int n,
                                   int nrhs,
                                   const mcComplex *const Aarray[],
                                   int lda,
                                   const int *devIpiv,
                                   mcComplex *const Barray[],
                                   int ldb,
                                   int *info,
                                   int batchSize);

mcblasStatus_t mcblasZgetrsBatched(mcblasHandle_t handle,
                                   mcblasOperation_t trans,
                                   int n,
                                   int nrhs,
                                   const mcDoubleComplex *const Aarray[],
                                   int lda,
                                   const int *devIpiv,
                                   mcDoubleComplex *const Barray[],
                                   int ldb,
                                   int *info,
                                   int batchSize);
```

该函数求解一组线性方程组，方程组格式如下：

$$\mathrm{op}(A[i])X[i] = \alpha \mathrm{B}[i]$$

其中 $A[i]$ 是一个已经过主元 LU 分解的矩阵，$X[i]$ 和 $B[i]$ 是 $n \times nrhs$ 矩阵。此外，对于矩阵 A

$$op(A[i]) = \begin{cases} A[i] & if\ transa == MCBLAS\_OP\_N \\ A^T[i] & if\ transa == MCBLAS\_OP\_T \\ A^H[i] & if\ transa == MCBLAS\_OP\_C \end{cases}$$

此函数用于较小的矩阵，其中启动开销是一个重要因素。

如果 devIpiv 为 nil，则 mcblas<t>getrsBatched 支持非主元 LU 分解。

mcblas<t>getrsBatched 支持任意维度。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
trans		input	非转置或共轭运算 op(A)
n		input	Aarray[i] 的行数和列数
nrhs		input	Barray[i] 的列数
Aarray	设备	input	指向 <type> 数组的指针数组，每个数组维度为 n x n，其中 lda >=max(1,n)
lda		input	用于存储矩阵 Aarray[i] 的二维数组的前导维度
devIpiv	设备	input	大小为 n x batchSize 的数组，包含以线性方式存储的 Aarray[i] 每个因式分解的旋转序列。如果 devIpiv 为零，则忽略所有 Aarray[i] 的旋转
Barray	设备	in/out	指向 <type> 数组的指针数组，每个数组维度为 n x nrhs，其中 ldb >=max(1,n) 。矩阵 Barray[i] 不应重叠；否则，将出现未定义的行为。
ldb		input	用于存储矩阵 Barray[i] 的二维数组的前导维度
info	主机	output	如果 info=0，执行成功。 如果 info = -j，第 j 个参数具有非法值。
batchSize		input	A 中包含的指针数

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_INVALID_VALUE	• n < 0 或 nrhs < 0 • trans != MCBLAS_OP_N, MCBLAS_OP_C, MCBLAS_OP_T • lda < max(1, n) • ldb < max(1, n)
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.6.5 mcblas<t>getriBatched()

```
mcblasStatus_t mcblasSgetriBatched(mcblasHandle_t handle,
                                   int n,
                                   const float *const Aarray[],
                                   int lda,
                                   int *PivotArray,
                                   float *const Carray[],
                                   int ldc,
                                   int *infoArray,
                                   int batchSize);

mcblasStatus_t mcblasDgetriBatched(mcblasHandle_t handle,
                                   int n,
                                   const double *const Aarray[],
                                   int lda,
```

(下页继续)

(续上页)

```
                                    int *PivotArray,
                                    double *const Carray[],
                                    int ldc,
                                    int *infoArray,
                                    int batchSize);

mcblasStatus_t mcblasCgetriBatched(mcblasHandle_t handle,
                                    int n,
                                    const mcComplex *const Aarray[],
                                    int lda,
                                    int *PivotArray,
                                    mcComplex *const Carray[],
                                    int ldc,
                                    int *infoArray,
                                    int batchSize);

mcblasStatus_t mcblasZgetriBatched(mcblasHandle_t handle,
                                    int n,
                                    const mcDoubleComplex *const Aarray[],
                                    int lda,
                                    int *PivotArray,
                                    mcDoubleComplex *const Carray[],
                                    int ldc,
                                    int *infoArray,
                                    int batchSize);
```

`Aarray` 和 `Carray` 是指向矩阵的指针数组，以列主格式存储，维度为 `n*n`，前导维度分别为 `lda` 和 `ldc`。

该函数执行 `A[i]` 矩阵反转，i = 0，…，`batchSize-1`。

在调用 mcblas<t>getriBatched 之前，必须先使用例程 mcblas<t>getrfBatched 对矩阵 `A[i]` 进行分解。在调用 mcblas<t>getrfBatched 后，`Aarray[i]` 指向的矩阵将包含矩阵 `A[i]` 的 LU 分解，`(PivotArray+i)` 指向的向量将包含主元序列。

LU 分解之后，mcblas<t>getriBatched 使用正向和反向三角求解器（triangular solver）来完成 `A[i]` 矩阵反转，i = 0，…，`batchSize-1`。矩阵反转为异地操作，因此 Carray[i] 的内存空间不能与 Array[i] 的内存空间重叠。

通常，mcblas<t>getrfBatched 中的所有参数都会传递到 mcblas<t>getriBatched 中。例如，

```
//第 1 步：执行就地 LU 分解，P*A = L*U
// Aarray[i] 是 n*n 矩阵 A[i]
      mcblasDgetrfBatched(handle, n, Aarray, lda, PivotArray, infoArray,
↪batchSize);
// 查看 infoArray[i] 以确认 A[i] 分解是否成功。
// Array[i] 包含 A[i] LU 分解

//第 2 步：执行异地反转，Carray[i] = inv(A[i])
      mcblasDgetriBatched(handle, n, Aarray, lda, PivotArray, Carray, ldc,
↪infoArray, batchSize);
// 查看 infoArray[i] 以确认 A[i] 反转是否成功。
```

用户可以从 mcblas<t>getrfBatched 或 mcblas<t>getriBatched 中检查奇异性。

此函数用于较小的矩阵，其中启动开销是一个重要因素。

如果对 mcblas<t>getrfBatched 执行非主元反转，则 mcblas<t>getriBatched 的 `PivotArray` 应为 nil。

mcblas<t>getriBatched 支持任意维度。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
n		input	Aarray[i] 的行数和列数
Aarray	设备	input	指向 <type> 数组的指针数组，每个数组维度为 n x n，其中 lda >=max(1,n)
lda		input	用于存储矩阵 Aarray[i] 的二维数组的前导维度
PivotArray	设备	output	大小为 `n x batchSize` 的数组，包含以线性方式存储的 Aarray[i] 每个因式分解的旋转序列。如果 PivotArray 为零，则禁用旋转
Carray	设备	output	指向 <type> 数组的指针数组，每个数组维度为 n x n，其中 ldc >=max(1,n) 。矩阵 Carray[i] 不应重叠；否则，将出现未定义的行为。
ldc		input	用于存储矩阵 Carray[i] 的二维数组的前导维度
infoArray	设备	output	大小为 `batchSize` 的数组，info(=infoArray[i]) 包含 `A` 的因式分解信息。 如果 info=0，执行成功。 如果 info = k，U(k,k) 为 0。U 恰好是奇异的，且反转失败。
batchSize		input	A 中包含的指针数

下面列出了此函数可能的返回值及其含义。

返回值	含义
`MCBLAS_STATUS_SUCCESS`	操作成功
`MCBLAS_STATUS_NOT_INITIALIZED`	库未初始化
`MCBLAS_STATUS_INVALID_VALUE`	• `n < 0,lda < 0,ldc < 0` 或 `batchSize < 0` • `lda < n` 或 `ldb < n`
`MCBLAS_STATUS_EXECUTION_FAILED`	此函数在 GPU 上启用失败

2.6.6 mcblas<t>matinvBatched()

```
mcblasStatus_t mcblasSmatinvBatched(mcblasHandle_t handle,
                                    int n,
                                    const float *const A[],
                                    int lda,
                                    float *const Ainv[],
                                    int lda_inv,
                                    int *info,
                                    int batchSize);

mcblasStatus_t mcblasDmatinvBatched(mcblasHandle_t handle,
                                    int n,
                                    const double *const A[],
                                    int lda,
                                    double *const Ainv[],
                                    int lda_inv,
                                    int *info,
                                    int batchSize);

mcblasStatus_t mcblasCmatinvBatched(mcblasHandle_t handle,
                                    int n,
                                    const mcComplex *const A[],
                                    int lda,
```

(下页继续)

(续上页)

```
                                    mcComplex *const Ainv[],
                                    int lda_inv,
                                    int *info,
                                    int batchSize);

mcblasStatus_t mcblasZmatinvBatched(mcblasHandle_t handle,
                                    int n,
                                    const mcDoubleComplex *const A[],
                                    int lda,
                                    mcDoubleComplex *const Ainv[],
                                    int lda_inv,
                                    int *info,
                                    int batchSize);
```

A 和 Ainv 是指向矩阵的指针数组，以列主格式存储，维度为 n*n，前导维度分别为 lda 和 lda_inv。

该函数执行 A[i] 矩阵反转，i = 0，…，batchSize-1。

此函数是 mcblas<t>getrfBatched 加 mcblas<t>getriBatched 的快捷方式。但如果 n 大于 32，则此函数无效。如果此函数无效，用户必须使用 mcblas<t>getrfBatched 和 mcblas<t>getriBatched。

如果 A[i] 是奇异矩阵，则 info[i] 报告奇异性，与 mcblas<t>getrfBatched 相同。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
n		input	A[i] 的行数和列数
A	设备	input	指向 <type> 数组的指针数组，每个数组维度为 n x n，其中 lda >=max(1,n)
lda		input	用于存储矩阵 A[i] 的二维数组的前导维度
Ainv	设备	output	指向 <type> 数组的指针数组，每个数组维度为 n x n，其中 lda >=max(1,n) 。矩阵 Ainv[i] 不应重叠；否则，将出现未定义的行为。
lda_inv		input	用于存储矩阵 Ainv[i] 的二维数组的前导维度
info	设备	output	大小为 batchSize 的数组，info[i] 包含 A[i] 的反转信息。 如果 info=0，执行成功。 如果 info = k，U(k,k) 为 0。U 恰好是奇异的，且反转失败。
batchSize		input	A 中包含的指针数

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_INVALID_VALUE	• n < 0，lda < 0，lda_inv < 0 或 batchSize < 0 • lda < n 或 lda_inv < n • n > 32
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.6.7 mcblas<t>geqrfBatched()

```
mcblasStatus_t mcblasSgeqrfBatched(mcblasHandle_t handle,
                                   int m,
                                   int n,
                                   float *const Aarray[],
                                   int lda,
                                   float *const TauArray[],
                                   int *info,
                                   int batchSize);

mcblasStatus_t mcblasDgeqrfBatched(mcblasHandle_t handle,
                                   int m,
                                   int n,
                                   double *const Aarray[],
                                   int lda,
                                   double *const TauArray[],
                                   int *info,
                                   int batchSize);

mcblasStatus_t mcblasCgeqrfBatched(mcblasHandle_t handle,
                                   int m,
                                   int n,
                                   mcComplex *const Aarray[],
                                   int lda,
                                   mcComplex *const TauArray[],
                                   int *info,
                                   int batchSize);

mcblasStatus_t mcblasZgeqrfBatched(mcblasHandle_t handle,
                                   int m,
                                   int n,
                                   mcDoubleComplex *const Aarray[],
                                   int lda,
                                   mcDoubleComplex *const TauArray[],
                                   int *info,
                                   int batchSize);
```

`Aarray` 是一个指向矩阵的指针数组，以列主格式存储，维度为 `m x n`，前导维度为 `lda`。`TauArray` 是一个指针数组，指向维度至少为 `max (1, min(m, n)` 的向量。

该函数使用豪斯霍尔德（Householder）反射对每个 `Aarray[i]` 执行 QR 分解，`i = 0, ..., batchSize-1`。每个 `Q[i]` 矩阵表示初等反射算子（elementary reflector）的积，并存储在每个 `Aarray[i]` 的下半部分，如下所示：

```
Q[j] = H[j][1] H[j][2] . . . H[j](k), where k = min(m,n).
```

每个 H[j][i] 的形式为

:: H[j][i] = I - tau[j] * v * v'

其中 tau[j] 是一个实标量，而 v 是一个实向量，v(1:i-1) = 0 且 v(i) = 1。v(i+1:m) 会存储在 Aarray[j][i+1:m,i] 中，tau 存储在 TauArray[j][i] 中。

此函数用于较小的矩阵，其中启动开销是一个重要因素。

mcblas<t>geqrfBatched 支持任意维度。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
m		input	Aarray[i] 的行数

下页继续

表 2.8 – 续上页

参数	内存	In/out	含义
n		input	Aarray[i] 的列数
Aarray	设备	input	指向 <type> 数组的指针数组，每个数组维度为 m x n，其中 lda >=max(1,m)
lda		input	用于存储矩阵 Aarray[i] 的二维数组的前导维度
TauArray	设备	output	指向 <type> 数组的指针数组，每个向量维度为 max (1,min(m,n))
info	设备	output	如果 info=0，则传递给函数的参数有效； 如果 info<0，则 postion-info 中的参数无效
batchSize		input	A 中包含的指针数

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_INVALID_VALUE	• If m < 0，n < 0 或 batchSize < 0 • lda < max(1, m)
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.6.8 mcblas<t>gelsBatched()

```
mcblasStatus_t mcblasSgelsBatched(mcblasHandle_t handle,
                                  mcblasOperation_t trans,
                                  int m,
                                  int n,
                                  int nrhs,
                                  float *const Aarray[],
                                  int lda,
                                  float *const Carray[],
                                  int ldc,
                                  int *info,
                                  int *devInfoArray,
                                  int batchSize );

mcblasStatus_t mcblasDgelsBatched(mcblasHandle_t handle,
                                  mcblasOperation_t trans,
                                  int m,
                                  int n,
                                  int nrhs,
                                  double *const Aarray[],
                                  int lda,
                                  double *const Carray[],
                                  int ldc,
                                  int *info,
                                  int *devInfoArray,
                                  int batchSize );

mcblasStatus_t mcblasCgelsBatched(mcblasHandle_t handle,
                                  mcblasOperation_t trans,
                                  int m,
                                  int n,
```

(下页继续)

(续上页)

```
                                    int nrhs,
                                    mcComplex *const Aarray[],
                                    int lda,
                                    mcComplex *const Carray[],
                                    int ldc,
                                    int *info,
                                    int *devInfoArray,
                                    int batchSize );

mcblasStatus_t mcblasZgelsBatched(mcblasHandle_t handle,
                                    mcblasOperation_t trans,
                                    int m,
                                    int n,
                                    int nrhs,
                                    mcDoubleComplex *const Aarray[],
                                    int lda,
                                    mcDoubleComplex *const Carray[],
                                    int ldc,
                                    int *info,
                                    int *devInfoArray,
                                    int batchSize );
```

`Aarray` 是一个指向矩阵的指针数组，以列主格式存储，维度为 `m x n`，前导维度为 `lda`。`Carray` 是一个指向矩阵的指针数组，以列主格式存储，维度为 `n x nrhs`，前导维度为 `ldc`。

该函数查找一组超定方程组（overdetermined systems）的最小二乘解（least squares solution），解决如下所述的最小二乘问题：

```
minimize  || Carray[i] - Aarray[i]*Xarray[i] ||  , with i = 0, ...,
↪batchSize-1
```

结束时，每个 `Aarray[i]` 都由其 QR 分解覆盖，每个 `Carray[i]` 都由最小二乘解覆盖。

mcblas<t>gelsBatched 仅支持非转置操作，并且只求解超定方程组（m >=n）。

此函数用于较小的矩阵，其中启动开销是一个重要因素。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
trans		input	非转置或共轭转置运算 op(Aarray[i])。目前仅支持非转置运算
m		input	Aarray[i] 的行数
n		input	每个 Aarray[i] 的列数和每个 Carray[i] 的行数
nrhs		input	每个 Carray[i] 的列数
Aarray	设备	in/out	指向 <type> 数组的指针数组。每个数组维度为 m x n，其中 lda >=max(1,m)。 矩阵 Aarray[i] 不应重叠；否则，将出现未定义的行为。
lda		input	用于存储矩阵 Aarray[i] 的二维数组的前导维度
Carray	设备	in/out	指向 <type> 数组的指针数组。每个数组维度为 n x nrhs，其中 ldc >=max(1,m)。 矩阵 Carray[i] 不应重叠；否则，将出现未定义的行为。
ldc		input	用于存储矩阵 Aarray[i] 的二维数组的前导维度
info	设备	output	如果 info=0，则传递给函数的参数有效； 如果 info<0，则 postion-info 中的参数无效

下页继续

表 2.9 – 续上页

参数	内存	In/out	含义
devInfoArray	设备	output	维度为 batchsize 的整数的可选数组。 如果不是 null，则每个元素 devInfoArray[i] 都包含一个值 V，其含义如下： V = 0：第 i 个问题已成功解决； V > 0：Aarray[i] 的第 V 个对角线元素为 0。数组 Aarray[i] 没有完整的秩。
batchSize		input	Aarray 和 Carray 中包含的指针数

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_INVALID_VALUE	• m < 0，n < 0，nrhs < 0 或 batchSize < 0 • lda < max(1, m) 或 ldc < max(1, m)
MCBLAS_STATUS_NOT_SUPPORTED	参数 m <n 或 trans 与非转置不同
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.6.9 mcblas<t>tpttr()

```
mcblasStatus_t mcblasStpttr(mcblasHandle_t handle,
                            mcblasFillMode_t uplo,
                            int n,
                            const float *AP,
                            float *A,
                            int lda );

mcblasStatus_t mcblasDtpttr(mcblasHandle_t handle,
                            mcblasFillMode_t uplo,
                            int n,
                            const double *AP,
                            double *A,
                            int lda );

mcblasStatus_t mcblasCtpttr(mcblasHandle_t handle,
                            mcblasFillMode_t uplo,
                            int n,
                            const mcComplex *AP,
                            mcComplex *A,
                            int lda );

mcblasStatus_t mcblasZtpttr(mcblasHandle_t handle,
                            mcblasFillMode_t uplo
                            int n,
                            const mcDoubleComplex *AP,
                            mcDoubleComplex *A,
                            int lda );
```

此函数执行从三角压缩格式转换到三角格式。

如果 uplo == MCBLAS_FILL_MODE_LOWER，则 AP 的元素将复制到三角矩阵 A 的下三角部分，而 A 的上三角部分保持不变。如果 uplo == MCBLAS_FILL_MODE_UPPER，则 AP 的元素将复制到三角矩阵 A 的上三角部分，而 A 的下三角部分保持不变。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
uplo		input	指示矩阵 AP 包含矩阵 A 的下半部分还是上半部分
n		input	矩阵 A 的行数和列数
AP	设备	input	以压缩格式存储 A 的 <type> 数组
A	设备	output	<type> 数组，维度为 lda x n，其中 lda>=max(1,n)。A 的另一侧不受影响。
lda		input	用于存储矩阵 A 的二维数组的前导维度

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_INVALID_VALUE	• n < 0 • uplo != MCBLAS_FILL_MODE_UPPER, MCBLAS_FILL_MODE_LOWER • lda < max(1, n)
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.6.10 mcblas<t>trttp()

```
mcblasStatus_t mcblasStrttp(mcblasHandle_t handle,
                            mcblasFillMode_t uplo,
                            int n,
                            const float *A,
                            int lda,
                            float *AP );

mcblasStatus_t mcblasDtrttp(mcblasHandle_t handle,
                            mcblasFillMode_t uplo,
                            int n,
                            const double *A,
                            int lda,
                            double *AP );

mcblasStatus_t mcblasCtrttp(mcblasHandle_t handle,
                            mcblasFillMode_t uplo,
                            int n,
                            const mcComplex *A,
                            int lda,
                            mcComplex *AP );

mcblasStatus_t mcblasZtrttp(mcblasHandle_t handle,
                            mcblasFillMode_t uplo,
                            int n,
                            const mcDoubleComplex *A,
                            int lda,
                            mcDoubleComplex *AP );
```

此函数执行从三角格式转换到三角压缩格式。

如果 uplo == MCBLAS_FILL_MODE_LOWER，则三角矩阵 A 的下三角部分将复制到 AP 数组中。如果 uplo == MCBLAS_FILL_MODE_UPPER，则三角矩阵 A 的上三角部分将复制到 AP 数组中。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
uplo		input	指示引用的是矩阵 A 的下半部分还是上半部分
n		input	矩阵 A 的行数和列数
A	设备	input	<type> 数组，维度为 lda x n，其中 lda>=max(1,n)。
lda		input	用于存储矩阵 A 的二维数组的前导维度
AP	设备	output	以压缩格式存储 A 的 <type> 数组

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_INVALID_VALUE	• n < 0 • uplo != MCBLAS_FILL_MODE_UPPER, MCBLAS_FILL_MODE_LOWER • lda < max(1, n)
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.6.11 mcblas<t>gemmEx()

```
mcblasStatus_t mcblasSgemmEx(mcblasHandle_t handle,
                             mcblasOperation_t transa,
                             mcblasOperation_t transb,
                             int m,
                             int n,
                             int k,
                             const float    *alpha,
                             const void     *A,
                             macaDataType_t Atype,
                             int lda,
                             const void     *B,
                             macaDataType_t Btype,
                             int ldb,
                             const float    *beta,
                             void           *C,
                             macaDataType_t Ctype,
                             int ldc)
mcblasStatus_t mcblasCgemmEx(mcblasHandle_t handle,
                             mcblasOperation_t transa,
                             mcblasOperation_t transb,
                             int m,
                             int n,
                             int k,
                             const mcComplex *alpha,
                             const void      *A,
                             macaDataType_t  Atype,
                             int lda,
                             const void      *B,
                             macaDataType_t  Btype,
                             int ldb,
                             const mcComplex *beta,
```

(下页继续)

(续上页)

```
                                void            *C,
                                macaDataType_t  Ctype,
                                int ldc)
```

此函数是 mcblas<t>gemm. 的扩展。在此函数中，输入矩阵和输出矩阵可以具有较低的精度，但计算仍然以 <t> 类型完成。例如，mcblasSgemmEx 的 float 类型，mcblasCgemmEx 的 mcComplex 类型。

$$C = \alpha \text{op}(A)\text{op}(B) + \beta C$$

其中 α 和 β 是标量，A、B、C 是以列主格式存储的矩阵，维度分别为 op(A) $m \times k$、op(B) $k \times n$ 和 C $m \times n$。此外，对于矩阵 A

对于矩阵 B，op(B) 的定义类似。.

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
transa		input	非转置或共轭转置运算 op(A)
transb		input	非转置或共轭转置运算 op(B)
m		input	矩阵 op(A) 和 C 的行数
n		input	矩阵 op(B) 和 C 的列数
k		input	op(A) 的列数和 op(B) 的行数
alpha	主机或设备	input	用于乘法的 <type> 标量。
A	设备	input	如果 transa == MCBLAS_OP_N，<type> 数组维度为 lda x k，其中 lda>=max(1,m)；否则，维度为 lda x m，其中 lda>=max(1,k)
Atype		input	枚举，指定矩阵 A 的数据类型
lda		input	用于存储矩阵 A 的二维数组的前导维度
B	设备	input	如果 transb == MCBLAS_OP_N，<type> 数组维度为 ldb x n，其中 ldb>=max(1,k)；否则，维度为 ldb x k，其中 ldb>=max(1,n)
Btype		input	枚举，指定矩阵 B 的数据类型
ldb		input	用于存储矩阵 B 的二维数组的前导维度
beta	主机或设备	input	用于乘法的 <type> 标量，如果 beta == 0，则 C 不必为有效输入
C	设备	in/out	<type> 数组，维度为 ldc x n 其中 ldc>=max(1,m)
Ctype		input	枚举，指定矩阵 C 的数据类型
ldc		input	用于存储矩阵 C 的二维数组的前导维度

mcblasSgemmEx 支持的矩阵类型组合如下所示：

C	A/B
MACA_R_16BF	MACA_R_16BF
MACA_R_16F	MACA_R_16F
MACA_R_32F	MACA_R_8I
	MACA_R_16BF
	MACA_R_16F
	MACA_R_32F

mcblasCgemmEx 支持的矩阵类型组合如下所示：

C	A/B
MACA_C_32F	MACA_C_8I
	MACA_C_32F

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_ARCH_MISMATCH	当前函数功能在本设备上不支持
MCBLAS_STATUS_NOT_SUPPORTED	不支持参数 Atype, Btype 和 Ctype 的组合
MCBLAS_STATUS_INVALID_VALUE	• m < 0，n < 0 或 k < 0 • transa 或 transb != MCBLAS_OP_N, MCBLAS_OP_C, MCBLAS_OP_T • 如果 transa == MCBLAS_OP_N ，lda < max(1, m)；否则 lda < max(1, k) • 如果 transb == MCBLAS_OP_N ，ldb < max(1, k)；否则 ldb < max(1, n) • ldc < max(1, m)
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.6.12 mcblasGemmEx()

```
mcblasStatus_t mcblasGemmEx(mcblasHandle_t handle,
                            mcblasOperation_t transa,
                            mcblasOperation_t transb,
                            int m,
                            int n,
                            int k,
                            const void    *alpha,
                            const void     *A,
                            macaDataType_t Atype,
                            int lda,
                            const void     *B,
                            macaDataType_t Btype,
                            int ldb,
                            const void    *beta,
                            void           *C,
                            macaDataType_t Ctype,
                            int ldc,
                            mcblasComputeType_t computeType,
                            mcblasGemmAlgo_t algo)

#if defined(__cplusplus)
mcblasStatus_t mcblasGemmEx(mcblasHandle_t handle,
                            mcblasOperation_t transa,
                            mcblasOperation_t transb,
                            int m,
                            int n,
                            int k,
                            const void     *alpha,
                            const void     *A,
                            macaDataType   Atype,
                            int lda,
                            const void     *B,
                            macaDataType   Btype,
                            int ldb,
                            const void     *beta,
                            void           *C,
```

(下页继续)

(续上页)

```
                                macaDataType    Ctype,
                                int ldc,
                                mcblasComputeType    computeType,
                                mcblasGemmAlgo_t algo)
#endif
```

此函数是 mcblas<t>gemm 的扩展，允许用户单独指定每个 A、B、C 矩阵的数据类型，计算精度和要运行的 GEMM 算法。本节进一步列出了支持的参数组合。

mcblasGemmEx 函数的第二个变量用于向后兼容 C++ 应用程序代码，其中 computeType 参数是 macaDataType，而不是 mcblasComputeType_t。C 应用程序仍将使用更新的函数签名进行编译。

$$C = \alpha \text{op}(A)\text{op}(B) + \beta C$$

其中 α 和 β 是标量，A、B、C 是以列主格式存储的矩阵，维度分别为 op(A) $m \times k$、op(B) $k \times n$ 和 C $m \times n$。此外，对于矩阵 A

$$op(A) = \begin{cases} A & if\ transa == MCBLAS\_OP\_N \\ A^T & if\ transa == MCBLAS\_OP\_T \\ A^H & if\ transa == MCBLAS\_OP\_C \end{cases}$$

对于矩阵 B，op(B) 的定义类似。.

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
transa		input	非转置或共轭转置运算 op(A)
transb		input	非转置或共轭转置运算 op(B)
m		input	矩阵 op(A) 和 C 的行数
n		input	矩阵 op(B) 和 C 的列数
k		input	op(A) 的列数和 op(B) 的行数
alpha	主机或设备	input	computeType 和 Ctype 对应类型的 A*B 的缩放因子，详情请参见下表
A	设备	input	如果 transa == MCBLAS_OP_N，<type> 数组维度为 lda x k，其中 lda>=max(1,m)；否则，维度为 lda x m，其中 lda>=max(1,k)
Atype		input	枚举，指定矩阵 A 的数据类型
lda		input	用于存储矩阵 A 的二维数组的前导维度
B	设备	input	如果 transb == MCBLAS_OP_N，<type> 数组维度为 ldb x n，其中 ldb>=max(1,k)；否则，维度为 ldb x k，其中 ldb>=max(1,n)
Btype		input	枚举，指定矩阵 B 的数据类型
ldb		input	用于存储矩阵 B 的二维数组的前导维度
beta	主机或设备	input	computeType 和 Ctype 对应类型的 C 的缩放因子，详情请参见下表。如果 beta == 0，则 C 不必为有效输入
C	设备	in/out	<type> 数组，维度为 ldc x n 其中 ldc>=max(1,m)
Ctype		input	枚举，指定矩阵 C 的数据类型
ldc		input	用于存储矩阵 C 的二维数组的前导维度
computeType		input	指定计算类型的枚举
algo		input	指定算法的枚举

mcblasGemmEx 支持以下计算类型，缩放类型，Atype/Btype 和 Ctype：

计算类型	缩放类型 (alpha and beta)	Atype/Btype	Ctype
MCBLAS_COMPUTE_16F or MCBLAS_COMPUTE_16F_PEDANTIC	MACA_R_16F	MACA_R_16F	MACA_R_16F
MCBLAS_COMPUTE_32I or MCBLAS_COMPUTE_32I_PEDANTIC	MACA_R_32I	MACA_R_8I	MACA_R_32I
MCBLAS_COMPUTE_32F or MCBLAS_COMPUTE_32F_PEDANTIC	MACA_R_32F	MACA_R_16BF	MACA_R_16BF
		MACA_R_16F	MACA_R_16F
		MACA_R_8I	MACA_R_32F
		MACA_R_16BF	MACA_R_32F
		MACA_R_16F	MACA_R_32F
		MACA_R_32F	MACA_R_32F
	MACA_C_32F	MACA_C_8I	MACA_C_32F
		MACA_C_32F	MACA_C_32F
MCBLAS_COMPUTE_32F_FAST_16F or MCBLAS_COMPUTE_32F_FAST_16BF or MCBLAS_COMPUTE_32F_FAST_TF32	MACA_R_32F	MACA_R_32F	MACA_R_32F
	MACA_C_32F	MACA_C_32F	MACA_C_32F
MCBLAS_COMPUTE_64F or MCBLAS_COMPUTE_64F_PEDANTIC	MACA_R_64F	MACA_R_64F	MACA_R_64F
	MACA_C_64F	MACA_C_64F	MACA_C_64F

注解： MCBLAS_COMPUTE_32I 和 MCBLAS_COMPUTE_32I_PEDANTIC 计算类型仅支持 A、B 为 4 字节对齐，lda、ldb 为 4 的倍数。

mcblasGemmEx 例程用于运行下表中的算法。

对于曦云系列 GPU，以下算法分别等同于 MCBLAS_GEMM_DEFAULT 或 MCBLAS_GEMM_DEFAULT_TENSOR_OP。为单精度操作指定算法 >= 99，等同于使用 MCBLAS_COMPUTE_32F_FAST_16F 计算类型，即使将数学模式或计算类型指定为 MCBLAS_COMPUTE_32F 或 MCBLAS_COMPUTE_32F_FAST_TF32。

mcblasGemmAlgo_t	含义
MCBLAS_GEMM_DEFAULT	应用启发式算法选择 GEMM 算法
MCBLAS_GEMM_ALGO0 to MCBLAS_GEMM_ALGO23	显式选择一个算法
MCBLAS_GEMM_DEFAULT_TENSOR_OP	应用启发式算法选择 GEMM 算法，同时尽可能允许使用 Tensor Core 操作
MCBLAS_GEMM_ALGO0_TENSOR_OP to MCBLAS_GEMM_ALGO15_TENSOR_OP	显式选择 GEMM 算法，并尽可能允许其使用 Tensor Core 操作，否则回退到基于 computeType 的 mcblas<t>gemmBatched

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_ARCH_MISMATCH	当前函数功能在本设备上不支持
MCBLAS_STATUS_NOT_SUPPORTED	不支持参数 Atype, Btype 和 Ctype 的组合，或算法 algo
MCBLAS_STATUS_INVALID_VALUE	• m < 0 , n < 0 或 k < 0 • transa 或 transb != MCBLAS_OP_N, MCBLAS_OP_C, MCBLAS_OP_T • 如果 transa == MCBLAS_OP_N , lda < max(1, m) ; 否则 lda < max(1, k) • 如果 transb == MCBLAS_OP_N , ldb < max(1, k) ; 否则 ldb < max(1, n) • ldc < max(1, m) • 不支持 Atype , Btype , Ctype 或 algo
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.6.13 mcblasGemmBatchedEx()

```
mcblasStatus_t mcblasGemmBatchedEx(mcblasHandle_t handle,
                                   mcblasOperation_t transa,
                                   mcblasOperation_t transb,
                                   int m,
                                   int n,
                                   int k,
                                   const void    *alpha,
                                   const void    *Aarray[],
                                   macaDataType_t Atype,
                                   int lda,
                                   const void    *Barray[],
                                   macaDataType_t Btype,
                                   int ldb,
                                   const void    *beta,
                                   void          *Carray[],
                                   macaDataType_t Ctype,
                                   int ldc,
                                   int batchCount,
                                   mcblasComputeType_t computeType,
                                   mcblasGemmAlgo_t algo)


#if defined(__cplusplus)
mcblasStatus_t mcblasGemmBatchedEx(mcblasHandle_t handle,
                                mcblasOperation_t transa,
                                mcblasOperation_t transb,
                                int m,
                                int n,
                                int k,
                                const void    *alpha,
                                const void    *Aarray[],
                                macaDataType   Atype,
                                int lda,
                                const void    *Barray[],
```

(下页继续)

(续上页)

```
                                macaDataType   Btype,
                                int ldb,
                                const void      *beta,
                                void            *Carray[],
                                macaDataType   Ctype,
                                int ldc,
                                int batchCount,
                                macaDataType   computeType,
                                mcblasGemmAlgo_t algo)
#endif
```

此函数是 mcblas<t>gemmBatched 的扩展，执行一批矩阵的矩阵-矩阵乘法，并允许用户单独指定每个 A、B、C 矩阵数组的数据类型，计算精度和要运行的 GEMM 算法。与 mcblas<t>gemmBatched 相似，该批是“均匀”的，即所有实例对于各自的 A、B、C 矩阵具有相同的维度（m，n，k），前导维度（lda，ldb，ldc）和转置（transa，transb）。批处理的每个实例的输入矩阵和输出矩阵的地址都是从调用者传递给函数的指针数组中读取的。本节进一步列出了支持的参数组合。

注解： mcblasGemmBatchedEx 函数的第二个变量用于向后兼容 C++ 应用程序代码，其中 computeType 参数是 macaDataType，而不是 mcblasComputeType_t。C 应用程序仍将使用更新的函数签名进行编译。

$$C[i] = \alpha \text{op}(A[i])\text{op}(B[i]) + \beta C[i], \text{ for i } \in [0, batchCount - 1]$$

其中 α 和 β 是标量，A、B、C 是指针数组，指向以列主格式存储的矩阵，维度分别为 op$(A[i])$ $m \times k$，op$(B[i])$ $k \times n$ 和 $C[i]$ $m \times n$。此外，对于矩阵 A

对于矩阵 $B[i]$，op$(B[i])$ 的定义类似。

注解： $C[i]$ 矩阵不能重叠，即单个 gemm 操作必须独立计算；否则，会导致未定义行为。在某些 problem size 上，可能需要在不同的 MXMACA 流中多次调用 mcblas<t>gemm，而不是使用此 API。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
transa		input	非转置或共轭转置运算 op([i])
transb		input	非转置或共轭转置运算 op(B[i])
m		input	矩阵 op(A[i]) 和 C[i] 的行数
n		input	矩阵 op(B[i]) 和 C[i] 的列数
k		input	op(A[i]) 的列数和 op(B[i]) 的行数
alpha	主机或设备	input	computeType 和 Ctype 对应类型的 A*B 的缩放因子，详情请参见下表
Aarray	设备	input	指向 <Atype> 数组的指针数组。如果 transa == MCBLAS_OP_N，数组维度为 lda x k，其中 lda>=max(1,m)；否则，维度为 lda x m，其中 lda>=max(1,k)。 所有指针都必须满足特定的对齐条件。详情参见以下内容
Atype		input	枚举，指定矩阵 Aarray 的数据类型
lda		input	用于存储矩阵 A[i] 的二维数组的前导维度

下页继续

表 2.11 – 续上页

参数	内存	In/out	含义
Barray	设备	input	指向 <Btype> 数组的指针数组。如果 transb == MCBLAS_OP_N，数组维度为 ldb x n，其中 ldb>=max(1,k)；否则，维度为 ldb x k，其中 ldb>=max(1,n)。 所有指针都必须满足特定的对齐条件。详情参见以下内容
Btype		input	枚举，指定矩阵 Barray 的数据类型
ldb		input	用于存储矩阵 B[i] 的二维数组的前导维度
beta	主机或设备	input	computeType 和 Ctype 对应类型的 C 的缩放因子，详情请参见下表。如果 beta == 0，则 C[i] 不必为有效输入
Carray	设备	in/out	指向 <Ctype> 数组的指针数组。维度为 ldc x n，其中 ldc>=max(1,m)。矩阵 C[i] 不应重叠；否则，将出现未定义的行为。 所有指针都必须满足特定的对齐条件。详情参见以下内容
Ctype		input	枚举，指定矩阵 Carray 的数据类型
ldc		input	用于存储每个矩阵 C[i] 的二维数组的前导维度
batchCount		input	Aarray、Barray 和 Carray 中包含的指针数
computeType		input	指定计算类型的枚举
algo		input	指定算法的枚举

mcblasGemmBatchedEx 支持以下计算类型，缩放类型，Atype/Btype 和 Ctype：

计算类型	缩放类型 (alpha and beta)	Atype/Btype	Ctype
MCBLAS_COMPUTE_16F or MCBLAS_COMPUTE_16F_PEDANTIC	MACA_R_16F	MACA_R_16F	MACA_R_16F
MCBLAS_COMPUTE_32I or MCBLAS_COMPUTE_32I_PEDANTIC	MACA_R_32I	MACA_R_8I	MACA_R_32I
MCBLAS_COMPUTE_32F or MCBLAS_COMPUTE_32F_PEDANTIC	MACA_R_32F	MACA_R_16BF	MACA_R_16BF
		MACA_R_16F	MACA_R_16F
		MACA_R_8I	MACA_R_32F
		MACA_R_16BF	MACA_R_32F
		MACA_R_16F	MACA_R_32F
		MACA_R_32F	MACA_R_32F
	MACA_C_32F	MACA_C_8I	MACA_C_32F
		MACA_C_32F	MACA_C_32F
MCBLAS_COMPUTE_32F_FAST_16F or MCBLAS_COMPUTE_32F_FAST_16BF or MCBLAS_COMPUTE_32F_FAST_TF32	MACA_R_32F	MACA_R_32F	MACA_R_32F
	MACA_C_32F	MACA_C_32F	MACA_C_32F
MCBLAS_COMPUTE_64F or MCBLAS_COMPUTE_64F_PEDANTIC	MACA_R_64F	MACA_R_64F	MACA_R_64F
	MACA_C_64F	MACA_C_64F	MACA_C_64F

如果 Atype 是 MACA_R_16F 或 MACA_R_16BF；或者 computeType 是任意 FAST 选项；或者当数学模式或 algo 启用快速数学模式时，GPU 内存中的指针（而非指针数组）必须正确对齐，以避免未对齐的内存访问错误。理想情况下，所有指针都至少对齐 16 个字节。否则，建议符合以下规则：

- 如果 k%8==0，则确保 intptr_t(ptr) % 16 == 0
- 如果 k%2==0，则确保 intptr_t(ptr) % 4 == 0

MCBLAS_COMPUTE_32I 和 MCBLAS_COMPUTE_32I_PEDANTIC 计算类型仅支持所有 A[i]、B[i] 指针为 4 字节对齐，lda、ldb 为 4 的倍数。为了获得更好的性能，建议满足 IMMA 内核对常规数据排序的要求。

mcblasGemmAlgo_t	含义
MCBLAS_GEMM_DEFAULT	应用启发式算法选择 GEMM 算法
MCBLAS_GEMM_ALGO0 to MCBLAS_GEMM_ALGO23	显式选择一个算法
MCBLAS_GEMM_DEFAULT_TENSOR_OP	应用启发式算法选择 GEMM 算法，同时尽可能允许使用 Tensor Core 操作
MCBLAS_GEMM_ALGO0_TENSOR_OP to MCBLAS_GEMM_ALGO15_TENSOR_OP	显式选择 GEMM 算法，并尽可能允许其使用 Tensor Core 操作，否则回退到基于 computeType 的 mcblas<t>gemmBatched

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_ARCH_MISMATCH	当前函数功能在本设备上不支持
MCBLAS_STATUS_NOT_SUPPORTED	不支持参数 Atype, Btype 和 Ctype 的组合，或算法 algo
MCBLAS_STATUS_INVALID_VALUE	• m < 0 , n < 0 或 k < 0 • transa 或 transb != MCBLAS_OP_N, MCBLAS_OP_C, MCBLAS_OP_T • 如果 transa == MCBLAS_OP_N , lda < max(1, m) ; 否则 lda < max(1, k) • 如果 transb == MCBLAS_OP_N , ldb < max(1, k) ; 否则 ldb < max(1, n) • ldc < max(1, m) • 不支持 Atype , Btype , Ctype , algo 或 computeType
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.6.14 mcblasGemmStridedBatchedEx()

```
mcblasStatus_t mcblasGemmStridedBatchedEx(mcblasHandle_t handle,
                                          mcblasOperation_t transa,
                                          mcblasOperation_t transb,
                                          int m,
                                          int n,
                                          int k,
                                          const void    *alpha,
                                          const void     *A,
                                          macaDataType_t Atype,
                                          int lda,
                                          mcblas_stride strideA,
                                          const void     *B,
                                          macaDataType_t Btype,
                                          int ldb,
                                          mcblas_stride strideB,
                                          const void    *beta,
                                          void           *C,
                                          macaDataType_t Ctype,
                                          int ldc,
                                          mcblas_stride strideC,
```

(下页继续)

(续上页)

```
                                        int batchCount,
                                        mcblasComputeType_t computeType,
                                        mcblasGemmAlgo_t algo)

#if defined(__cplusplus)
mcblasStatus_t mcblasGemmStridedBatchedEx(mcblasHandle_t handle,
                                        mcblasOperation_t transa,
                                        mcblasOperation_t transb,
                                        int m,
                                        int n,
                                        int k,
                                        const void    *alpha,
                                        const void     *A,
                                        macaDataType Atype,
                                        int lda,
                                        mcblas_stride strideA,
                                        const void     *B,
                                        macaDataType Btype,
                                        int ldb,
                                        mcblas_stride strideB,
                                        const void    *beta,
                                        void           *C,
                                        macaDataType Ctype,
                                        int ldc,
                                        mcblas_stride strideC,
                                        int batchCount,
                                        macaDataType computeType,
                                        mcblasGemmAlgo_t algo)
#endif
```

此函数是 mcblas<t>gemmStridedBatched 的扩展，执行一批矩阵的矩阵-矩阵乘法，并允许用户单独指定每个 A、B、C 矩阵的数据类型，计算精度和要运行的 GEMM 算法。与 mcblas<t>gemmStridedBatched 相似，该批是“均匀”的，即所有实例对于各自的 A、B、C 矩阵具有相同的维度（m，n，k），前导维度（lda，ldb，ldc）和转置（transa，transb）。批处理中每个实例的输入矩阵 A 和 B 以及输出矩阵 C 的位置，与前一个实例中的位置在元素数量上有固定的偏移。第一个实例中指向矩阵 A、B、C 的指针，与元素数量的偏移量（strideA、strideB、strideC）一并由用户传入函数，这些偏移量决定了后续实例中输入矩阵和输出矩阵的位置。

注解： mcblasGemmStridedBatchedEx 函数的第二个变量用于向后兼容 C++ 应用程序代码，其中 computeType 参数是 macaDataType_t，而不是 mcblasComputeType_t。C 应用程序仍将使用更新的函数签名进行编译。

$$C + i * strideC = \alpha \text{op}(A + i * strideA)\text{op}(B + i * strideB) + \beta(C + i * strideC),$$
$$\text{for i } \in [0, batchCount - 1]$$

其中 α 和 β 是标量，A、B、C 是指针数组，指向以列主格式存储的矩阵，维度分别为 op$(A[i])$ $m \times k$，op$(B[i])$ $k \times n$ 和 $C[i]$ $m \times n$。此外，对于矩阵 A

$$op(A) = \begin{cases} A & if\ transa == MCBLAS\_OP\_N \\ A^T & if\ transa == MCBLAS\_OP\_T \\ A^H & if\ transa == MCBLAS\_OP\_C \end{cases}$$

对于矩阵 $B[i]$，op$(B[i])$ 的定义类似。

注解： $C[i]$ 矩阵不能重叠，即单个 gemm 操作必须独立计算；否则，会导致未定义行为。在某些 problem

size 上，可能需要在不同的 MXMACA 流中多次调用 mcblas<t>gemm，而不是使用此 API。

在下表中，我们使用 A[i], B[i], C[i] 分别表示批处理的第 i 个实例中 A、B、C 矩阵，隐式假设它们分别是相对于 A[i-1], B[i-1], C[i-1] 的元素数量的偏移 strideA, strideB, strideC。偏移的单位是元素数量，不能为零。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
transa		input	非转置或共轭转置运算 op(A[i])
transb		input	非转置或共轭转置运算 op(B[i])
m		input	矩阵 op(A[i]) 和 C[i] 的行数
n		input	矩阵 op(B[i]) 和 C[i] 的列数
k		input	op(A[i]) 的列数和 op(B[i]) 的行数
alpha	主机或设备	input	computeType 和 Ctype 对应类型的 A*B 的缩放因子，详情请参见下表
A	设备	input	指向批处理第一个实例对应的 <Atype> 矩阵 A 的指针，如果 transa==MCBLAS_OP_N，维度为 lda x k，其中 lda>=max(1,m)；否则，维度为 lda x m，其中 lda>=max(1,k)
Atype		input	枚举，指定矩阵 A 的数据类型
lda		input	用于存储矩阵 A[i] 的二维数组的前导维度
strideA		input	long long int 类型的值，该值给出 A[i] 和 A[i+1] 之间的元素数偏移量
B	设备	input	指向批处理第一个实例对应的 <Btype> 矩阵 B 的指针，如果 transb==MCBLAS_OP_N，维度为 ldb x n，其中 ldb>=max(1,k)；否则，维度为 ldb x k，其中 ldb>=max(1,n)
Btype		input	枚举，指定矩阵 B 的数据类型
ldb		input	用于存储矩阵 B[i] 的二维数组的前导维度
strideB		input	long long int 类型的值，该值给出 B[i] 和 B[i+1] 之间的元素数偏移量
beta	主机或设备	input	computeType 和 Ctype 对应类型的 C 的缩放因子，详情请参见下表。如果 beta == 0，C[i] 不必为有效输入
C	设备	in/out	指向批处理第一个实例对应的 <Ctype> 矩阵 C 的指针，维度为 ldc x n，其中 ldc>=max(1,m)。矩阵 C[i] 不应重叠；否则，将出现未定义的行为。
Ctype		input	枚举，指定矩阵 C 的数据类型
ldc		input	用于存储矩阵 C[i] 的二维数组的前导维度
strideC		input	long long int 类型的值，该值给出 C[i] 和 C[i+1] 之间的元素数偏移量
batchCount		input	批处理中要执行的 GEMM 数量
computeType		input	指定计算类型的枚举
algo		input	指定算法的枚举

mcblasGemmStridedBatchedEx 支持以下计算类型，缩放类型，Atype/Btype 和 Ctype：

计算类型	缩放类型 (alpha and beta)	Atype/Btype	Ctype
MCBLAS_COMPUTE_16F or MCBLAS_COMPUTE_16F_PEDANTIC	MACA_R_16F	MACA_R_16F	MACA_R_16F
MCBLAS_COMPUTE_32I or MCBLAS_COMPUTE_32I_PEDANTIC	MACA_R_32I	MACA_R_8I	MACA_R_32I
MCBLAS_COMPUTE_32F or MCBLAS_COMPUTE_32F_PEDANTIC	MACA_R_32F	MACA_R_16BF	MACA_R_16BF
		MACA_R_16F	MACA_R_16F
		MACA_R_8I	MACA_R_32F
		MACA_R_16BF	MACA_R_32F
		MACA_R_16F	MACA_R_32F
		MACA_R_32F	MACA_R_32F
	MACA_C_32F	MACA_C_8I	MACA_C_32F
		MACA_C_32F	MACA_C_32F
MCBLAS_COMPUTE_32F_FAST_16F or MCBLAS_COMPUTE_32F_FAST_16BF or MCBLAS_COMPUTE_32F_FAST_TF32	MACA_R_32F	MACA_R_32F	MACA_R_32F
	MACA_C_32F	MACA_C_32F	MACA_C_32F
MCBLAS_COMPUTE_64F or MCBLAS_COMPUTE_64F_PEDANTIC	MACA_R_64F	MACA_R_64F	MACA_R_64F
	MACA_C_64F	MACA_C_64F	MACA_C_64F

MCBLAS_COMPUTE_32I 和 MCBLAS_COMPUTE_32I_PEDANTIC 计算类型仅支持所有 A[i]、B[i] 指针为 4 字节对齐，lda、ldb 为 4 的倍数。

mcblasGemmAlgo_t	含义
MCBLAS_GEMM_DEFAULT	应用启发式算法选择 GEMM 算法
MCBLAS_GEMM_ALGO0 to MCBLAS_GEMM_ALGO23	显式选择一个算法
MCBLAS_GEMM_DEFAULT_TENSOR_OP	应用启发式算法选择 GEMM 算法，同时尽可能允许使用 Tensor Core 操作
MCBLAS_GEMM_ALGO0_TENSOR_OP to MCBLAS_GEMM_ALGO15_TENSOR_OP	显式选择 GEMM 算法，并尽可能允许其使用 Tensor Core 操作，否则回退到基于 computeType 的 mcblas<t>gemmBatched

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_ARCH_MISMATCH	当前函数功能在本设备上不支持
MCBLAS_STATUS_NOT_SUPPORTED	不支持参数 Atype, Btype 和 Ctype 的组合，或算法 algo
MCBLAS_STATUS_INVALID_VALUE	• m < 0，n < 0 或 k < 0 • transa 或 transb != MCBLAS_OP_N, MCBLAS_OP_C, MCBLAS_OP_T • 如果 transa == MCBLAS_OP_N，lda < max(1, m)；否则 lda < max(1, k) • 如果 transb == MCBLAS_OP_N，ldb < max(1, k)；否则 ldb < max(1, n) • ldc < max(1, m) • 不支持 Atype，Btype，Ctype，algo 或 computeType
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.6.15 mcblasCsyrkEx()

```
mcblasStatus_t mcblasCsyrkEx(mcblasHandle_t handle,
                             mcblasFillMode_t uplo,
                             mcblasOperation_t trans,
                             int n,
                             int k,
                             const float     *alpha,
                             const void      *A,
                             macaDataType    Atype,
                             int lda,
                             const float    *beta,
                             mcComplex      *C,
                             macaDataType   Ctype,
                             int ldc)
```

此函数是 mcblasCsyrk 的扩展，其输入矩阵和输出矩阵可以具有较低的精度，但计算仍然以 mcComplex 类型完成。

此函数执行对称秩- k 校正

$$C = \alpha \text{op}(A) \text{op}(A)^T + \beta C$$

其中 α 和 β 是标量，C 是以 lower 或 upper 模式存储的对称矩阵，A 是一个维度为 op(A) $n \times k$ 的矩阵。此外，对于矩阵 A

$$op(A) = \begin{cases} A & if\ \ transa == MCBLAS\_OP\_N \\ A^T & if\ \ transa == MCBLAS\_OP\_T \end{cases}$$

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
uplo		input	指示存储的是矩阵 C 的上半部分还是下半部分，其对称部分未被引用，且是从存储的元素推断出来的
trans		input	非转置或转置运算 op(A)
n		input	矩阵 op(A) 和 C 的行数
k		input	矩阵 op(A) 的列数
alpha	主机或设备	input	用于乘法的 <type> 标量
A	设备	input	如果 trans==MCBLAS_OP_N，<type> 数组维度为 lda x k，其中 lda>=max(1,n)；否则，维度为 lda x n，其中 lda>=max(1,k)
Atype		input	枚举，指定矩阵 A 的数据类型
lda		input	用于存储矩阵 A 的二维数组的前导维度
beta	主机或设备	input	用于乘法的 <type> 标量，如果 beta == 0，则 C 不必为有效输入
C	设备	in/out	<type> 数组，维度为 ldc x n，其中 ldc>=max(1,n)。
Ctype		input	枚举，指定矩阵 C 的数据类型
ldc		input	用于存储矩阵 C 的二维数组的前导维度

mcblasCsyrkEx 支持的矩阵类型组合如下所示：

A	C
MACA_C_8I	MACA_C_32F
MACA_C_32F	MACA_C_32F

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_INVALID_VALUE	• n < 0 或 k < 0 • uplo != MCBLAS_FILL_MODE_UPPER, MCBLAS_FILL_MODE_LOWER • trans != MCBLAS_OP_N, MCBLAS_OP_C, MCBLAS_OP_T • 如果 trans == MCBLAS_OP_N，lda < max(1, n)；否则 lda < max(1, k) • ldc < max(1, n) • 不支持 Atype 或 Ctype
MCBLAS_STATUS_NOT_SUPPORTED	不支持参数 Atype 和 Ctype 的组合
MCBLAS_STATUS_ARCH_MISMATCH	当前函数功能在本设备上不支持
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.6.16 mcblasCsyrk3mEx()

```
mcblasStatus_t mcblasCsyrk3mEx(mcblasHandle_t handle,
                               mcblasFillMode_t uplo,
                               mcblasOperation_t trans,
                               int n,
                               int k,
                               const float     *alpha,
                               const void      *A,
                               macaDataType     Atype,
                               int lda,
                               const float     *beta,
                               mcComplex       *C,
                               macaDataType    Ctype,
                               int ldc)
```

此函数是 mcblasCsyrk 的扩展，其输入矩阵和输出矩阵可以具有较低的精度，但计算仍然以 mcComplex 类型完成。该例程使用高斯复杂度归约算法来实现，可使性能提升高达 25%。

此函数执行对称秩- k 校正

$$C = \alpha \text{op}(A) \text{op}(A)^T + \beta C$$

其中 α 和 β 是标量，C 是以 lower 或 upper 模式存储的对称矩阵，A 是一个维度为 op(A) $n \times k$ 的矩阵。此外，对于矩阵 A

$$op(A) = \begin{cases} A & if\ transa == MCBLAS\_OP\_N \\ A^T & if\ transa == MCBLAS\_OP\_T \end{cases}$$

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
uplo		input	指示存储的是矩阵 C 的上半部分还是下半部分，其对称部分未被引用，且是从存储的元素推断出来的
trans		input	非转置或转置运算 op(A)
n		input	矩阵 op(A) 和 C 的行数
k		input	矩阵 op(A) 的列数
alpha	主机或设备	input	用于乘法的 <type> 标量
A	设备	input	如果 trans==MCBLAS_OP_N，<type> 数组维度为 lda x k，其中 lda>=max(1,n)；否则，维度为 lda x n，其中 lda>=max(1,k)
Atype		input	枚举，指定矩阵 A 的数据类型
lda		input	用于存储矩阵 A 的二维数组的前导维度
beta	主机或设备	input	用于乘法的 <type> 标量，如果 beta == 0，则 C 不必为有效输入
C	设备	in/out	<type> 数组，维度为 ldc x n，其中 ldc>=max(1,n)。
Ctype		input	枚举，指定矩阵 C 的数据类型
ldc		input	用于存储矩阵 C 的二维数组的前导维度

mcblasCsyrk3mEx 支持的矩阵类型组合如下所示：

A	C
MACA_C_8I	MACA_C_32F
MACA_C_32F	MACA_C_32F

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_INVALID_VALUE	• n < 0 或 k < 0 • uplo != MCBLAS_FILL_MODE_UPPER, MCBLAS_FILL_MODE_LOWER • trans != MCBLAS_OP_N, MCBLAS_OP_C, MCBLAS_OP_T • 如果 trans == MCBLAS_OP_N，lda < max(1, n)；否则 lda < max(1, k) • ldc < max(1, n) • 不支持 Atype 或 Ctype
MCBLAS_STATUS_NOT_SUPPORTED	不支持参数 Atype 和 Ctype 的组合
MCBLAS_STATUS_ARCH_MISMATCH	当前函数功能在本设备上不支持
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.6.17 mcblasCherkEx()

```
mcblasStatus_t mcblasCherkEx(mcblasHandle_t handle,
                             mcblasFillMode_t uplo,
                             mcblasOperation_t trans,
                             int n,
                             int k,
                             const float     *alpha,
```

(下页继续)

(续上页)

```
                                const void      *A,
                                macaDataType    Atype,
                                int lda,
                                const float     *beta,
                                mcComplex       *C,
                                macaDataType    Ctype,
                                int ldc)
```

此函数是 mcblasCherk 的扩展，其输入矩阵和输出矩阵可以具有较低的精度，但计算仍然以 mcComplex 类型完成。

此函数执行厄米秩-k 校正

$$C = \alpha \text{op}(A) \text{op}(A)^H + \beta C$$

其中 α 和 β 是标量，C 是以 lower 或 upper 模式存储的厄米矩阵，A 是一个维度为 op(A) $n \times k$ 的矩阵。此外，对于矩阵 A

$$op(A) = \begin{cases} A & if\ transa == MCBLAS\_OP\_N \\ A^H & if\ transa == MCBLAS\_OP\_C \end{cases}$$

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
uplo		input	指示存储的是矩阵 A 的上半部分还是下半部分，另外 Hermitian 部分未被引用，且是从存储的元素推断出来的
trans		input	非转置或共轭转置运算 op(A)
n		input	矩阵 op(A) 和 C 的行数
k		input	矩阵 op(A) 的列数
alpha	主机或设备	input	用于乘法的 <type> 标量
A	设备	input	如果 transa==MCBLAS_OP_N，<type> 数组维度为 lda x k，其中 lda>=max(1,n)；否则，维度为 lda x n，其中 lda>=max(1,k)
Atype		input	枚举，指定矩阵 A 的数据类型
lda		input	用于存储矩阵 A 的二维数组的前导维度
beta	主机或设备	input	用于乘法的 <type> 标量，如果 beta == 0，则 C 不必为有效输入
C	设备	in/out	<type> 数组，维度为 ldc x n，其中 ldc>=max(1,n)。假定对角线元素的虚数部分设置为 0
Ctype		input	枚举，指定矩阵 C 的数据类型
ldc		input	用于存储矩阵 C 的二维数组的前导维度

mcblasCherkEx 支持的矩阵类型组合如下所示：

A	C
MACA_C_8I	MACA_C_32F
MACA_C_32F	MACA_C_32F

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_INVALID_VALUE	• n < 0 或 k < 0 • uplo != MCBLAS_FILL_MODE_UPPER, MCBLAS_FILL_MODE_LOWER • trans != MCBLAS_OP_N, MCBLAS_OP_C, MCBLAS_OP_T • 如果 trans == MCBLAS_OP_N，lda < max(1, n)；否则 lda < max(1, k) • ldc < max(1, n) • 不支持 Atype 或 Ctype
MCBLAS_STATUS_ARCH_MISMATCH	当前函数功能在本设备上不支持
MCBLAS_STATUS_NOT_SUPPORTED	不支持参数 Atype 和 Ctype 的组合
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.6.18 mcblasCherk3mEx()

```
mcblasStatus_t mcblasCherk3mEx(mcblasHandle_t handle,
                               mcblasFillMode_t uplo,
                               mcblasOperation_t trans,
                               int n,
                               int k,
                               const float     *alpha,
                               const void      *A,
                               macaDataType     Atype,
                               int lda,
                               const float     *beta,
                               mcComplex       *C,
                               macaDataType    Ctype,
                               int ldc)
```

此函数是 mcblasCherk 的扩展，其输入矩阵和输出矩阵可以具有较低的精度，但计算仍然以 mcComplex 类型完成。该例程使用高斯复杂度归约算法来实现，可使性能提升高达 25%。

此函数执行厄米秩- k 校正

$$C = \alpha \text{op}(A)\text{op}(A)^H + \beta C$$

其中 α 和 β 是标量，C 是以 lower 或 upper 模式存储的厄米矩阵，A 是一个维度为 op(A) $n \times k$ 的矩阵。此外，对于矩阵 A

$$op(A) = \begin{cases} A & if\ transa == MCBLAS\_OP\_N \\ A^H & if\ transa == MCBLAS\_OP\_C \end{cases}$$

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
uplo		input	指示存储的是矩阵 A 的上半部分还是下半部分，另外 Hermitian 部分未被引用，且是从存储的元素推断出来的
trans		input	非转置或共轭转置运算 op(A)
n		input	矩阵 op(A) 和 C 的行数
k		input	矩阵 op(A) 的列数
alpha	主机或设备	input	用于乘法的 <type> 标量

下页继续

表 2.12 – 续上页

参数	内存	In/out	含义
A	设备	input	如果 transa==MCBLAS_OP_N，<type> 数组维度为 lda x k，其中 lda>=max(1,n)；否则，维度为 lda x n，其中 lda>=max(1,k)
Atype		input	枚举，指定矩阵 A 的数据类型
lda		input	用于存储矩阵 A 的二维数组的前导维度
beta	主机或设备	input	用于乘法的 <type> 标量，如果 beta == 0，则 C 不必为有效输入
C	设备	in/out	<type> 数组，维度为 ldc x n，其中 ldc>=max(1,n)。假定对角线元素的虚数部分设置为 0
Ctype		input	枚举，指定矩阵 C 的数据类型
ldc		input	用于存储矩阵 C 的二维数组的前导维度

mcblasCherk3mEx 支持的矩阵类型组合如下所示：

A	C
MACA_C_8I	MACA_C_32F
MACA_C_32F	MACA_C_32F

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_INVALID_VALUE	• n < 0 或 k < 0 • uplo != MCBLAS_FILL_MODE_UPPER, MCBLAS_FILL_MODE_LOWER • trans != MCBLAS_OP_N, MCBLAS_OP_C, MCBLAS_OP_T • 如果 trans == MCBLAS_OP_N，lda < max(1, n)；否则 lda < max(1, k) • ldc < max(1, n) • 不支持 Atype 或 Ctype
MCBLAS_STATUS_NOT_SUPPORTED	不支持参数 Atype 和 Ctype 的组合
MCBLAS_STATUS_ARCH_MISMATCH	当前函数功能在本设备上不支持
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.6.19 mcblasNrm2Ex()

```
mcblasStatus_t  mcblasNrm2Ex(mcblasHandle_t handle,
                             int n,
                             const void *x,
                             macaDataType xType,
                             int incx,
                             void *result,
                             macaDataType resultType,
                             macaDataType executionType)
```

此函数是 mcblas<t>nrm2 的 API 泛化，可在其中独立指定输入数据，输出数据和计算类型。

该函数计算向量 x 的欧几里德范数（Euclidean norm）。代码中使用多相模型累积来避免中间结果下溢和溢出，其结果相当于精确运算中的 $\sqrt{\sum_{i=1}^{n}(\mathbf{x}[j]\times\mathbf{x}[j])}$，其中 $j = 1 + (i-1) * \text{incx}$。请注意，最后一个公式反映了基于 1 的索引，该索引用于与 Fortran 兼容。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
n		input	向量 x 中的元素数
x	设备	input	包含 n 个元素的 <type> 向量
xType		input	枚举，指定向量 x 的数据类型
incx		input	x 中连续元素之间的步幅
result	主机或设备	output	结果范数，如果 n,incx<=0，则为 0.0
resultType		input	枚举，指定 result 的数据类型
executionType		input	枚举，指定计算执行中的数据类型

mcblasNrm2Ex 支持的数据类型组合如下所示：

x	result	execution
MACA_R_16F	MACA_R_16F	MACA_R_32F
MACA_R_32F	MACA_R_32F	MACA_R_32F
MACA_R_64F	MACA_R_64F	MACA_R_64F
MACA_C_32F	MACA_C_32F	MACA_C_32F
MACA_C_64F	MACA_C_64F	MACA_C_64F

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_ALLOC_FAILED	无法分配归约缓冲区
MCBLAS_STATUS_NOT_SUPPORTED	不支持参数 xtype, resultType 和 executionType 的组合
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败
MCBLAS_STATUS_INVALID_VALUE	• 不支持参数 xtype, resultType 或 executionType • result == NULL

2.6.20 mcblasAxpyEx()

```
mcblasStatus_t mcblasAxpyEx(mcblasHandle_t handle,
                            int n,
                            const void *alpha,
                            macaDataType alphaType,
                            const void *x,
                            macaDataType xType,
                            int incx,
                            void *y,
                            macaDataType yType,
                            int incy,
                            macaDataType executiontype);
```

此函数是 mcblas<t>axpy 的 API 泛化，可在其中独立指定输入数据，输出数据和计算类型。

该函数将向量 x 乘以标量 α，并将其添加到 y 向量中，用结果覆盖最新向量。因此，执行的操作为 $\mathbf{y}[j] = \alpha \times \mathbf{x}[k] + \mathbf{y}[j]$，其中 $i = 1, \ldots, n$，$k = 1 + (i-1) * \text{incx}$，$j = 1 + (i-1) * \text{incy}$。请注意，最后两个公式反映了基于 1 的索引，该索引用于与 Fortran 兼容。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
alpha	主机或设备	input	用于乘法的 <type> 标量
n		input	向量 x 和 y 中的元素数
x	设备	input	包含 n 个元素的 <type> 向量
xType		input	枚举，指定向量 x 的数据类型
incx		input	x 中连续元素之间的步幅
y	设备	in/out	包含 n 个元素的 <type> 向量
yType		input	枚举，指定向量 y 的数据类型
incy		input	y 中连续元素之间的步幅
executionType		input	枚举，指定计算执行中的数据类型

mcblasAxpyEx 支持的数据类型组合如下所示：

x	y	execution
MACA_R_16F	MACA_R_16F	MACA_R_32F
MACA_R_32F	MACA_R_32F	MACA_R_32F
MACA_R_64F	MACA_R_64F	MACA_R_64F
MACA_C_32F	MACA_C_32F	MACA_C_32F
MACA_C_64F	MACA_C_64F	MACA_C_64F

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_NOT_SUPPORTED	不支持参数 xtype, yType 和 executionType 的组合
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败
MCBLAS_STATUS_INVALID_VALUE	不支持参数 alphaType，xType，yType 或 executionType

2.6.21 mcblasDotEx()

```
mcblasStatus_t mcblasDotEx(mcblasHandle_t handle,
                           int n,
                           const void *x,
                           macaDataType xType,
                           int incx,
                           const void *y,
                           macaDataType yType,
                           int incy,
                           void *result,
                           macaDataType resultType,
                           macaDataType executionType);

mcblasStatus_t mcblasDotcEx(mcblasHandle_t handle,
                           int n,
                           const void *x,
                           macaDataType xType,
                           int incx,
                           const void *y,
                           macaDataType yType,
```

(下页继续)

(续上页)

```
                                int incy,
                                void *result,
                                macaDataType resultType,
                                macaDataType executionType);
```

这些函数是 mcblas<t>dot 和 mcblas<t>dotc 的 API 泛化，可在其中独立指定输入数据，输出数据和计算类型。

注解: mcblas<t>dotc 是共轭点积，mcblas<t>dotu 是未共轭点积。

该函数计算向量 x 和 y 的点积（dot product）。因此，结果为 $\sum_{i=1}^{n}(\mathbf{x}[k] \times \mathbf{y}[j])$，其中 $k = 1+(i-1)*$incx，$j = 1+(i-1)*$ incy。请注意，在第一个方程中，如果函数名称以字符 “c” 结尾，则应使用 x 向量元素的共轭。最后两个公式反映了基于 1 的索引，该索引用于与 Fortran 兼容。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
n		input	向量 x 和 y 中的元素数
x	设备	input	包含 n 个元素的 <type> 向量
xType		input	枚举，指定向量 x 的数据类型
incx		input	x 中连续元素之间的步幅
y	设备	input	包含 n 个元素的 <type> 向量
yType		input	枚举，指定向量 y 的数据类型
incy		input	y 中连续元素之间的步幅
result	主机或设备	output	结果点积，如果 n<=0，则为 0.0
resultType		input	枚举，指定 result 的数据类型
executionType		input	枚举，指定计算执行中的数据类型

mcblasDotEx 和 mcblasDotcEx 支持的数据类型组合如下所示：

x	y	result	execution
MACA_R_16F	MACA_R_16F	MACA_R_16F	MACA_R_32F
MACA_R_32F	MACA_R_32F	MACA_R_32F	MACA_R_32F
MACA_R_64F	MACA_R_64F	MACA_R_64F	MACA_R_64F
MACA_C_32F	MACA_C_32F	MACA_C_32F	MACA_C_32F
MACA_C_64F	MACA_C_64F	MACA_C_64F	MACA_C_64F

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_ALLOC_FAILED	无法分配归约缓冲区
MCBLAS_STATUS_NOT_SUPPORTED	不支持参数 xtype，yType，resultType，和 executionType 的组合
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败
MCBLAS_STATUS_INVALID_VALUE	不支持参数 xType，yType，resultType 或 executionType

2.6.22 mcblasRotEx()

```
mcblasStatus_t mcblasRotEx(mcblasHandle_t handle,
                           int n,
                           void *x,
                           macaDataType xType,
                           int incx,
                           void *y,
                           macaDataType yType,
                           int incy,
                           const void *c,  /* host or device pointer */
                           const void *s,
                           macaDataType csType,
                           macaDataType executiontype);
```

此函数是 mcblas<t>rot 的扩展，可在其中独立指定输入数据，输出数据，cosine/sine 类型和计算类型。

该函数将吉文斯旋转 (Givens rotation) 矩阵 (即，在 x，y 平面上逆时针旋转，旋转角度由 cos(alpha)=c，sin(alpha)=s 定义)：

$G = \begin{pmatrix} c & s \\ -s & c \end{pmatrix}$ 应用到向量 x 和 y 中。

因此，结果为 $\mathbf{x}[k] = c \times \mathbf{x}[k] + s \times \mathbf{y}[j]$ 和 $\mathbf{y}[j] = -s \times \mathbf{x}[k] + c \times \mathbf{y}[j]$，其中 $k = 1 + (i-1) * \text{incx}$，$j = 1 + (i-1) * \text{incy}$。请注意，最后两个公式反映了基于 1 的索引，该索引用于与 Fortran 兼容。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
n		input	向量 x 和 y 中的元素数
x	设备	in/out	包含 n 个元素的 <type> 向量
xType		input	枚举，指定向量 x 的数据类型
incx		input	x 中连续元素之间的步幅
y	设备	in/out	包含 n 个元素的 <type> 向量
yType		input	枚举，指定向量 y 的数据类型
incy		input	y 中连续元素之间的步幅
c	主机或设备	output	旋转矩阵余弦元素
c	主机或设备	output	旋转矩阵正弦元素
csType		input	枚举，指定 c 和 s 的数据类型
executionType		input	枚举，指定计算执行中的数据类型

mcblasRotEx 支持的数据类型组合如下所示：

executionType	xType / yType	csType
MACA_R_32F	MACA_R_16BF	MACA_R_16BF
	MACA_R_16F	MACA_R_16F
	MACA_R_32F	MACA_R_32F
MACA_R_64F	MACA_R_64F	MACA_R_64F
MACA_C_32F	MACA_C_32F	MACA_R_32F
	MACA_C_32F	MACA_C_32F
MACA_C_64F	MACA_C_64F	MACA_R_64F
	MACA_C_64F	MACA_C_64F

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败

2.6.23 mcblasScalEx()

```
mcblasStatus_t  mcblasScalEx(mcblasHandle_t handle,
                             int n,
                             const void *alpha,
                             macaDataType alphaType,
                             void *x,
                             macaDataType xType,
                             int incx,
                             macaDataType executionType);
```

该函数按标量 α 缩放向量 x，并用结果覆盖该向量。因此，执行的操作为 $\mathbf{x}[j] = \alpha \times \mathbf{x}[j]$，其中 $i = 1, \ldots, n$，$j = 1 + (i-1) * \text{incx}$。请注意，最后两个公式反映了基于 1 的索引，该索引用于与 Fortran 兼容。

参数	内存	In/out	含义
handle		input	mcBLAS 库上下文的句柄
alpha	主机或设备	input	用于乘法的 <type> 标量
n		input	向量 x 中的元素数
x	设备	in/out	包含 n 个元素的 <type> 向量
xType		input	枚举，指定向量 x 的数据类型
incx		input	x 中连续元素之间的步幅
executionType		input	枚举，指定计算执行中的数据类型

mcblasScalEx 支持的数据类型组合如下所示：

x	execution
MACA_R_16F	MACA_R_32F
MACA_R_32F	MACA_R_32F
MACA_R_64F	MACA_R_64F
MACA_C_32F	MACA_C_32F
MACA_C_64F	MACA_C_64F

下面列出了此函数可能的返回值及其含义。

返回值	含义
MCBLAS_STATUS_SUCCESS	操作成功
MCBLAS_STATUS_NOT_INITIALIZED	库未初始化
MCBLAS_STATUS_NOT_SUPPORTED	不支持参数 xtype 和 executionType 的组合
MCBLAS_STATUS_EXECUTION_FAILED	此函数在 GPU 上启用失败
MCBLAS_STATUS_INVALID_VALUE	不支持参数 alphaType, xType, 或 executionType