# 曦云® 系列通用计算 GPU

## mcSolverIT API 参考

CSRD-23018-020-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 介绍

mcSolverIT 库是建立在沐曦 MXMACA® 运行时环境 (MXMACA 工具包的一部分) 和沐曦软件库之上实现的稀疏线性系统迭代求解器和预条件子。该库设计成可从 C++ 代码中调用。它使用沐曦的 MXMACA 运行时环境在 GPU 设备上运行。

mcSolverIT 库支持以下求解器或预条件子：

- **代数多网格 (AMG)**：提供经典的 AMG 和聚合 AMG，其中包含 AMG 求解过程的不同选项。例如，循环可以是 CG，CGF，F，V 或 W。
- **简单的松弛迭代求解器**：例如，高斯-赛戴尔 (Gauss-Seidel)、雅可比 (Jacobi) 及其变体。
- **多项式迭代求解器**：例如，切比雪夫多项式 (Chebyshev Polynomial) 方法。
- **克雷洛夫求解器**：例如，共轭梯度 (Conjugate Gradient，CG)、广义最小残差 (Generalized Minimum Residual，GMRES) 及其变体。
- **不完全直接求解器**：例如，不完全 LU 分解。
- **预条件求解器**：例如，预条件共轭梯度 (Preconditioned Conjugate Gradient，PCG)、预条件广义最小残差 (Preconditioned Generalized Minimum Residual，PGMRES) 等。

mcSolverIT 的功能分为以下几类：

- **实用程序参考**：描述用于内存管理、文件输入/输出以及链接库实用程序的可用实用程序函数。
- **配置参考**：描述配置句柄的操作，该句柄将包含内存信息和求解器设置。
- **资源参考**：描述资源句柄的操作，该句柄指示求解器将使用的内存资源。
- **矩阵参考**：描述矩阵句柄的操作。
- **向量参考**：描述向量句柄的操作。
- **求解器参考**：描述求解器句柄的操作。

## 1.1 安装 mcSolverIT

mcSolverIT 是一个与 MXMACA 工具包一起发布和安装的库。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 MXMACA 工具包后，请确保设置了环境变量 MACA_PATH。

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

mcSolverIT API 相关文件如下所示。

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

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

在使用 mcSolverIT 库编译代码之前，请确保环境变量 ISU_FASTMODEL 设置为 1。

```
export ISU_FASTMODEL=1
```

## 1.2 示例

请参考使用 mcSolverIT 编写代码的示例。它们展示了使用 mcSolverIT 库应用程序编程接口 (Application Programing Interface，API) 编写的一些 C++ 应用程序。一个单节点示例如下所示。

```
#include <vector>

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "amgx_c.h"
#include "mc_runtime.h"

int main() {
   // 步骤 1: 输入索引基数为 0 的 Csr 矩阵
   int n = 12;
   int nnz = 61;
   int block_dimx = 1;
   int block_dimy = 1;
   std::vector<int> A_rows = {0, 4, 8, 13, 21, 25, 32, 36, 41, 46, 50, 57,
↪61};
   std::vector<int> A_cols = {0, 1, 3, 8,
                              0, 1, 2, 3,
                              1, 2, 3, 4, 5,
                              0, 1, 2, 3, 4, 5, 8, 10,
                              2, 4, 5, 6,
                              2, 3, 4, 5, 6, 7, 10,
                              4, 5, 6, 7,
                              5, 6, 7, 9, 10,
                              0, 3, 8, 10, 11,
                              7, 9, 10, 11,
                              3, 5, 7, 8, 9, 10, 11,
                              8, 9, 10, 11};
   std::vector<double> A_vals = {1, 2, 3, 4,
                                 5, 6, 7, 8,
                                 9, 10, 11, 12, 13,
                                 14, 15, 16, 17, 18, 19, 20, 21,
                                 22, 23, 24, 25,
                                 26, 27, 28, 29, 30, 31, 32,
                                 33, 34, 35, 36,
                                 37, 38, 39, 40, 41,
                                 42, 43, 44, 45, 46,
                                 47, 48, 49, 50,
                                 51, 52, 53, 54, 55, 56, 57,
                                 58, 59, 60, 61};
   std::vector<double> h_x(n, 0);
   std::vector<double> h_b(n, 1);
   void *d_A_rows, *d_A_cols, *d_A_vals, *d_x, *d_b;
   mcMalloc((void **)&d_A_rows, (n + 1) * sizeof(int));
   mcMalloc((void **)&d_A_cols, nnz * sizeof(int));
   mcMalloc((void **)&d_A_vals, nnz * sizeof(double));
```

(下页继续)

(续上页)

```
    mcMalloc((void **)&d_x, n * sizeof(double));
    mcMalloc((void **)&d_b, n * sizeof(double));
    mcMemcpy(d_A_rows, A_rows.data(), (n + 1) * sizeof(int),
             mcMemcpyHostToDevice);
    mcMemcpy(d_A_cols, A_cols.data(), nnz * sizeof(int),
             mcMemcpyHostToDevice);
    mcMemcpy(d_A_vals, A_vals.data(), nnz * sizeof(double),
             mcMemcpyHostToDevice);
    mcMemcpy(d_x, h_x.data(), n * sizeof(double),
             mcMemcpyHostToDevice);
    mcMemcpy(d_b, h_b.data(), n * sizeof(double),
             mcMemcpyHostToDevice);

    // 步骤 2: 声明库句柄并初始化
    // 库句柄
    MCSOLVERIT_Mode mode = MCSOLVERIT_mode_dDDI;
    MCSOLVERIT_config_handle config_handle;
    MCSOLVERIT_resources_handle resources_handle;
    MCSOLVERIT_matrix_handle mtx_handle;
    MCSOLVERIT_vector_handle rhs_handle, sol_handle;
    MCSOLVERIT_solver_handle solver_handle;
    // 状态处理
    MCSOLVERIT_SOLVE_STATUS status;
    void *lib_handle = NULL;
    // 初始化
    MCSOLVERIT_initialize();
    MCSOLVERIT_initialize_plugins();

    // 步骤 3: 创建句柄
    MCSOLVERIT_config_create(&config_handle, "solver=AMG, cycle=V, \
                             monitor_residual=1, store_res_history=1, \
                             max_iters=10, tolerance=1e-6");
    MCSOLVERIT_resources_create_simple(&resources_handle, config_handle);
    MCSOLVERIT_solver_create(&solver_handle, resources_handle, mode, config_
↪handle);
    MCSOLVERIT_matrix_create(&mtx_handle, resources_handle, mode);
    MCSOLVERIT_vector_create(&rhs_handle, resources_handle, mode);
    MCSOLVERIT_vector_create(&sol_handle, resources_handle, mode);
    // 如果对角线包含在矩阵本身中，则 diag_data 必须设置为 NULL
    MCSOLVERIT_matrix_upload_all(mtx_handle, n, nnz, block_dimx, block_dimy,
                                 (int*)d_A_rows, (int*)d_A_cols, d_A_vals,
                                 NULL);
    MCSOLVERIT_vector_upload(rhs_handle, n, block_dimy, d_b);
    MCSOLVERIT_vector_upload(sol_handle, n, block_dimx, d_x);

    // 步骤 4: 求解器设置
    MCSOLVERIT_solver_setup(solver_handle, mtx_handle);

    // 步骤 5: 求解器求解
    MCSOLVERIT_solver_solve(solver_handle, rhs_handle, sol_handle);

    // 步骤 6: 获取求解器状态、迭代和残差
    MCSOLVERIT_solver_get_status(solver_handle, &status);
    int iterations = 0;
    double initial_residual;
    double final_residual;
```

(下页继续)

(续上页)

```
    MCSOLVERIT_solver_get_iterations_number(solver_handle, &iterations);
    MCSOLVERIT_solver_get_iteration_residual(solver_handle, 0, 0,
                                             &initial_residual);
    MCSOLVERIT_solver_get_iteration_residual(solver_handle, iterations, 0,
                                             &final_residual);
    final_residual = final_residual / initial_residual;
    printf("iterations = %d, relative residual = %le\n", iterations,
           final_residual);

    // 步骤 7: 检查结果
    double test_tolerance = 1e-04;
    if (status == MCSOLVERIT_SOLVE_FAILED) {
       printf("Error: Solver err\n");
    }
    if (test_tolerance < final_residual) {
       printf("Error: Solver not convergence\n");
    }

    // 步骤 8: 释放资源并退出
    // 销毁资源、矩阵、向量和求解器
    MCSOLVERIT_solver_destroy(solver_handle);
    MCSOLVERIT_vector_destroy(sol_handle);
    MCSOLVERIT_vector_destroy(rhs_handle);
    MCSOLVERIT_matrix_destroy(mtx_handle);
    MCSOLVERIT_resources_destroy(resources_handle);
    // 销毁配置
    MCSOLVERIT_config_destroy(config_handle);
    // 关机和退出
    MCSOLVERIT_finalize_plugins();
    MCSOLVERIT_finalize();
    // 释放资源
    mcFree(d_x);
    mcFree(d_b);
    mcFree(d_A_rows);
    mcFree(d_A_cols);
    mcFree(d_A_vals);

    return 0;
}
```

如果要使用其他求解器，可以在 MCSOLVERIT_config_create 中更改传递给配置句柄的设置，也可以将 MCSOLVERIT_config_create_from_file 与配置文件一起使用来设置求解器操作。

# 2 mcSolverIT 应用程序编程接口

## 2.1 类型参考

### 2.1.1 MCSOLVERIT_Mode

此类型指示求解器将在主机或设备上运行，以及数据和索引的精度。第一个字母表示求解器将在主机 (h) 或设备 (d) 上运行。第二个字母表示向量数据的精度是单精度浮点型 (F)、双精度浮点型 (D)、单精度复数型 (C) 还是双精度复数型 (Z)。第三个字母表示矩阵数据是精度是单精度浮点型 (F)、双精度浮点型 (D)、单精度复数型 (C) 还是双精度复数型 (Z)。最后一个字母表示所有索引的精度均为 32 位整型 (I)。它可以具有以下值

| 值 | 含义 |
| --- | --- |
| MCSOLVERIT_unset | 模式未设置 |
| MCSOLVERIT_modeRange | 最大 MCSOLVERIT_Mode 数 |
| MCSOLVERIT_mode_hDDI | • 求解器在主机上运行。<br>• 向量数据的精度是双精度浮点型。<br>• 向量数据的精度是双精度浮点型。<br>• 矩阵数据的精度是双精度浮点型。<br>• 所有指数的精度是 32 位整型。 |
| MCSOLVERIT_mode_hDFI | • 求解器在主机上运行。<br>• 向量数据的精度是双精度浮点型。<br>• 矩阵数据的精度是单精度浮点型<br>• 所有指数的精度是 32 位整型。 |
| MCSOLVERIT_mode_hFFI | • 求解器在主机上运行。<br>• 向量数据的精度是单精度浮点型。<br>• 矩阵数据的精度是单精度浮点型。<br>• 所有指数的精度是 32 位整型。 |
| MCSOLVERIT_mode_dDDI | • 求解器在设备上运行。<br>• 向量数据的精度是双精度浮点型。<br>• 矩阵数据的精度是双精度浮点型。<br>• 所有指数的精度是 32 位整型。 |

下页继续

表 2.1 – 续上页

| 值 | 含义 |
| --- | --- |
| MCSOLVERIT_mode_dDFI | • 求解器在设备上运行。<br>• 向量数据的精度是双精度浮<br>• 矩阵数据的精度是单精度浮点型。<br>• 所有指数的精度是 32 位整型。 |
| MCSOLVERIT_mode_dFFI | • 求解器在设备上运行。<br>• 向量数据的精度是单精度浮点型。<br>• 矩阵数据的精度是单精度浮点型。<br>• 所有指数的精度是 32 位整型。 |
| MCSOLVERIT_mode_hIDI | • 求解器在主机上运行。<br>• 向量数据的精度是 32 位整型。<br>• 矩阵数据的精度是双精度浮点型。<br>• 所有指数的精度是 32 位整型。 |
| MCSOLVERIT_mode_hIFI | • 求解器在主机上运行。<br>• 向量数据的精度是 32 位整型。<br>• 矩阵数据的精度是单精度浮点型。<br>• 所有指数的精度是 32 位整型。 |
| MCSOLVERIT_mode_dIDI | • 求解器在设备上运行。<br>• 向量数据的精度是 32 位整型。<br>• 矩阵数据的精度是双精度浮点型。<br>• 所有指数的精度是 32 位整型。 |
| MCSOLVERIT_mode_dIFI | • 求解器在设备上运行。<br>• 向量数据的精度是 32 位整型。<br>• 矩阵数据的精度是单精度浮点型。<br>• 所有指数的精度是 32 位整型。 |
| MCSOLVERIT_mode_hZZI | • 求解器在主机上运行。<br>• 向量数据的精度是双精度复数型。<br>• 矩阵数据的精度是双精度复数型。<br>• 所有指数的精度是 32 位整型。 |
| MCSOLVERIT_mode_hZCI | • 求解器在主机上运行。<br>• 向量数据的精度是双精度复数型。<br>• 矩阵数据的精度是单精度复数型。<br>• 所有指数的精度是 32 位整型。 |
| MCSOLVERIT_mode_hCCI | • 求解器在主机上运行。<br>• 向量数据的精度是单精度复数型。<br>• 矩阵数据的精度是单精度复数型。<br>• 所有指数的精度是 32 位整型。 |

下页继续

表 2.1 – 续上页

| 值 | 含义 |
| --- | --- |
| MCSOLVERIT_mode_dZZI | • 求解器在设备上运行。<br>• 矩阵数据的精度是单精度复数型。<br>• 矩阵数据的精度是单精度复数型。<br>• 所有指数的精度是 32 位整型。 |
| MCSOLVERIT_mode_dZCI | • 求解器在设备上运行。<br>• 向量数据的精度是双精度复数型。<br>• 矩阵数据的精度是单精度复数型。<br>• 所有指数的精度是 32 位整型。 |
| MCSOLVERIT_mode_dCCI | • 求解器在设备上运行。<br>• 矩阵数据的精度是单精度复数型。<br>• 矩阵数据的精度是单精度复数型。<br>• 所有指数的精度是 32 位整型。 |
| MCSOLVERIT_modeNum | |
| MCSOLVERIT_ModeInst | |

### 2.1.2 MCSOLVERIT_RC

此数据类型表示库函数返回的状态。值如下所示。

| 值 | 含义 |
| --- | --- |
| MCSOLVERIT_RC_OK | 没有错误。 |
| MCSOLVERIT_RC_BAD_PARAMETERS | amgx 调用的参数不正确。 |
| MCSOLVERIT_RC_UNKNOWN | 未知错误。 |
| MCSOLVERIT_RC_NOT_SUPPORTED_TARGET | 不支持的设备/主机算法。 |
| MCSOLVERIT_RC_NOT_SUPPORTED_BLOCKSIZE | 算法不支持的块大小。 |
| MCSOLVERIT_RC_MACA_FAILURE | MACA 内核启动错误。 |
| MCSOLVERIT_RC_THRUST_FAILURE | Thrust 故障。 |
| MCSOLVERIT_RC_NO_MEMORY | 内存不足。 |
| MCSOLVERIT_RC_IO_ERROR | 输入/输出错误。 |
| MCSOLVERIT_RC_BAD_MODE | C API 模式不正确。 |
| MCSOLVERIT_RC_CORE | 初始化 amgx 内核时出错。 |
| MCSOLVERIT_RC_PLUGIN | 初始化插件时出错。 |
| MCSOLVERIT_RC_BAD_CONFIGURATION | 提供的 amgx 配置不正确。 |
| MCSOLVERIT_RC_NOT_IMPLEMENTED | 配置功能未实现。 |
| MCSOLVERIT_RC_LICENSE_NOT_FOUND | 未找到有效的许可证。 |
| MCSOLVERIT_RC_INTERNAL | 内部错误。 |

### 2.1.3 MCSOLVERIT_SOLVE_STATUS

此数据类型表示求解阶段的状态，可以通过 MCSOLVERIT_solver_get_status 获取。值如下所示。

| 值 | 含义 |
| --- | --- |
| MCSOLVERIT_SOLVE_SUCCESS | 求解器收敛，没有错误 |
| MCSOLVERIT_SOLVE_FAILED | 求解器由于某种原因失败了 |
| MCSOLVERIT_SOLVE_DIVERGED | 求解器未失败，但达到了最大迭代次数并且残差未收敛 |
| MCSOLVERIT_SOLVE_NOT_CONVERGED | 求解器未达到收敛状态 |

## 2.2 实用程序参考

### 2.2.1 MCSOLVERIT_initialize()

```
MCSOLVERIT_RC MCSOLVERIT_API MCSOLVERIT_initialize()
```

初始化库。

### 2.2.2 MCSOLVERIT_API MCSOLVERIT_finalize()

```
MCSOLVERIT_RC MCSOLVERIT_API MCSOLVERIT_finalize()
```

完成库。

### 2.2.3 MCSOLVERIT_initialize_plugins()

```
MCSOLVERIT_RC MCSOLVERIT_API MCSOLVERIT_initialize_plugins()
```

初始化库的内置插件。

### 2.2.4 MCSOLVERIT_finalize_plugins()

```
MCSOLVERIT_RC MCSOLVERIT_API MCSOLVERIT_finalize_plugins()
```

完成库的内置插件。

### 2.2.5 MCSOLVERIT_get_api_version()

```
MCSOLVERIT_RC MCSOLVERIT_API MCSOLVERIT_get_api_version(int *major, int
↪*minor)
```

获取库的 API 版本。

**输入**

| 参数 | 内存 | 输入/输出 | 含义 |
| --- | --- | --- | --- |
| major | HOST | OUTPUT | 主要 API 版本号 |
| minor | HOST | OUTPUT | 次要 API 版本号 |

### 2.2.6 MCSOLVERIT_get_error_string()

```
MCSOLVERIT_RC MCSOLVERIT_API MCSOLVERIT_get_error_string(
                                    MCSOLVERIT_RC err,
                                    char *buf,
                                    int buf_len)
```

从错误代码中获取错误字符串。

**输入**

| 参数 | 内存 | 输入/输出 | 含义 |
|---|---|---|---|
| err | HOST | INPUT | 错误代码 |
| buf_len | HOST | INPUT | 字符串缓冲区的长度 |

**输出**

| 参数 | 内存 | 输入/输出 | 含义 |
|---|---|---|---|
| buf | HOST | OUTPUT | 字符串缓冲区，用于获取错误字符串 |

### 2.2.7 MCSOLVERIT_get_build_info_strings()

```
MCSOLVERIT_RC MCSOLVERIT_API MCSOLVERIT_get_build_info_strings(
                                    char **version,
                                    char **date,
                                    char **time)
```

获取有关库的构建信息。

**输出**

| 参数 | 内存 | 输入/输出 | 含义 |
|---|---|---|---|
| version | HOST | OUTPUT | 构建版本 |
| date | HOST | OUTPUT | 构建日期 |
| time | HOST | OUTPUT | 构建时间 |

### 2.2.8 MCSOLVERIT_pin_Memory()

```
MCSOLVERIT_RC MCSOLVERIT_API MCSOLVERIT_pin_Memory(
                                    void *ptr,
                                    unsigned int bytes)
```

通知操作系统将缓冲区固定在内存中。

**输入**

| 参数 | 内存 | 输入/输出 | 含义 |
|---|---|---|---|
| ptr | HOST | INPUT | 要固定的缓冲区 |
| bytes | HOST | INPUT | 要固定的字节数 |

### 2.2.9 MCSOLVERIT_unpin__Memory()

```
MCSOLVERIT_RC MCSOLVERIT_API MCSOLVERIT_unpin__Memory(void *ptr)
```

通知操作系统取消固定缓冲区。

**输入**

| 参数 | 内存 | 输入/输出 | 含义 |
| --- | --- | --- | --- |
| ptr | HOST | INPUT | 要取消固定的缓冲区 |

### 2.2.10 MCSOLVERIT_install_signal_handler()

```
MCSOLVERIT_RC MCSOLVERIT_API MCSOLVERIT_install_signal_handler()
```

安装默认的信号处理程序。

### 2.2.11 MCSOLVERIT_reset_signal_handler()

```
MCSOLVERIT_RC MCSOLVERIT_API MCSOLVERIT_reset_signal_handler()
```

恢复到先前的信号处理程序。

### 2.2.12 MCSOLVERIT_register_print_callback()

```
MCSOLVERIT_RC MCSOLVERIT_API MCSOLVERIT_register_print_callback(MCSOLVERIT_
↪print_callback func)
```

注册一个由用户提供的回调函数来处理文本输出。

**输入**

| 参数 | 内存 | 输入/输出 | 含义 |
| --- | --- | --- | --- |
| func | HOST | INPUT | 要注册的回调函数 |

### 2.2.13 MCSOLVERIT_read_system()

```
MCSOLVERIT_RC MCSOLVERIT_API MCSOLVERIT_read_system(
                                MCSOLVERIT_matrix_handle mtx,
                                MCSOLVERIT_vector_handle rhs,
                                MCSOLVERIT_vector_handle sol,
                                const char *filename)
```

从文件中读取线性系统。

**输入**

| 参数 | 内存 | 输入/输出 | 含义 |
| --- | --- | --- | --- |
| filename | HOST | INPUT | 要读取的文件 |

**输出**

| 参数 | 内存 | 输入/输出 | 含义 |
|---|---|---|---|
| mtx | HOST | OUTPUT | 系数矩阵的句柄 |
| rhs | HOST | OUTPUT | 右侧向量的句柄 |
| sol | HOST | OUTPUT | 解向量的句柄 |

### 2.2.14 MCSOLVERIT_read_system_distributed()

```
MCSOLVERIT_RC MCSOLVERIT_API MCSOLVERIT_read_system_distributed(
                                    MCSOLVERIT_matrix_handle mtx,
                                    MCSOLVERIT_vector_handle rhs,
                                    MCSOLVERIT_vector_handle sol,
                                    const char *filename,
                                    int allocated_halo_depth,
                                    int num_partitions,
                                    const int *partition_sizes,
                                    int partition_vector_size,
                                    const int *partition_vector)
```

从文件中读取线性系统的一个子集。

**输入**

| 参数 | 内存 | 输入/输出 | 含义 |
|---|---|---|---|
| filename | HOST | INPUT | 要读取的文件 |
| allocated_halo_depth | HOST | INPUT | 用于 halo 交换的深度，必须等于 num_import_rings |
| num_partitions | HOST | INPUT | 分区数 |
| partition_sizes | HOST | INPUT | 大小为 num_partitions 的数组，存储每个分区的大小。如果为 NULL，则会从分区向量中计算得到 |
| partition_vector_size | HOST | INPUT | partition_vector 的长度 |
| partition_vector | HOST | INPUT | 全局矩阵的分区分配 |

**输出**

| 参数 | 内存 | 输入/输出 | 含义 |
|---|---|---|---|
| mtx | HOST | OUTPUT | 系数矩阵的句柄 |
| rhs | HOST | OUTPUT | 右侧向量的句柄 |
| sol | HOST | OUTPUT | 解向量的句柄 |

### 2.2.15 MCSOLVERIT_read_system_global()

```
MCSOLVERIT_RC MCSOLVERIT_API MCSOLVERIT_read_system_global(
                                    int *n,
                                    int *nnz,
                                    int *block_dimx,
                                    int *block_dimy,
                                    int **row_ptrs,
                                    void **col_indices_global,
                                    void **data,
                                    void **diag_data,
                                    void **rhs,
```

(下页继续)

(续上页)

```
                                    void **sol,
                                    MCSOLVERIT_resources_handle rsc,
                                    MCSOLVERIT_Mode mode,
                                    const char *filename,
                                    int allocated_halo_depth,
                                    int num_partitions,
                                    const int *partition_sizes,
                                    int partition_vector_size,
                                    const int *partition_vector)
```

从文件中读取线性系统的一个子集，内部分配的 C 缓冲区（使用全局列索引）。**输入**

| 参数 | 内存 | 输入/输出 | 含义 |
|---|---|---|---|
| rsc | HOST | INPUT | 存储库资源的句柄 |
| mode | HOST | INPUT | 求解器模式 |
| filename | HOST | INPUT | 要读取的文件 |
| allocated_halo_depth | HOST | INPUT | 用于 halo 交换的深度，现在必须等于 num_import_rings |
| num_partitions | HOST | INPUT | 分区数 |
| partition_sizes | HOST | INPUT | 大小为 num_partitions 的数组，存储每个分区的大小。如果为 NULL，则会从分区向量中计算得到 |
| partition_vector_size | HOST | INPUT | partition_vector 的长度 |
| partition_vector | HOST | INPUT | 全局矩阵的分区分配 |

**输出**

| 参数 | 内存 | 输入/输出 | 含义 |
|---|---|---|---|
| n | HOST | OUTPUT | 局部矩阵的维度 |
| nnz | HOST | OUTPUT | 非零块的数量 |
| block_dimx | HOST | OUTPUT | 块的 x 维度 |
| block_dimy | HOST | OUTPUT | 块的 y 维度 |
| row_ptrs | HOST | OUTPUT | 指向全局矩阵中所有局部矩阵索引的指针 |
| col_indices_global | HOST | OUTPUT | 指向全局矩阵中所有局部矩阵列索引的指针 |
| data | HOST | OUTPUT | 指向所有局部矩阵值的指针 |
| diag_data | HOST | OUTPUT | 指向所有局部矩阵每行的外部对角线元素的指针。如果对角线数据包含在矩阵数据中，则设置为 NULL |
| rhs | HOST | OUTPUT | 右侧向量的句柄 |
| sol | HOST | OUTPUT | 解向量的句柄 |

### 2.2.16 MCSOLVERIT_read_system_maps_one_ring()

```
MCSOLVERIT_RC MCSOLVERIT_API MCSOLVERIT_read_system_maps_one_ring(
                                    int *n,
                                    int *nnz,
                                    int *block_dimx,
                                    int *block_dimy,
                                    int **row_ptrs,
                                    int **col_indices,
                                    void **data,
```

(下页继续)

(续上页)

```
                                    void **diag_data,
                                    void **rhs,
                                    void **sol,
                                    int *num_neighbors,
                                    int **neighbors,
                                    int **send_sizes,
                                    int ***send_maps,
                                    int **recv_sizes,
                                    int ***recv_maps,
                                    MCSOLVERIT_resources_handle rsc,
                                    MCSOLVERIT_Mode mode,
                                    const char *filename,
                                    int allocated_halo_depth,
                                    int num_partitions,
                                    const int *partition_sizes,
                                    int partition_vector_size,
                                    const int *partition_vector)
```

从文件中读取线性系统的一个子集，内部分配的 C 缓冲区（使用全局列索引）。

**输入**

| 参数 | 内存 | 输入/输出 | 含义 |
|---|---|---|---|
| rsc | HOST | INPUT | 存储库资源的句柄 |
| mode | HOST | INPUT | 求解器模式 |
| filename | HOST | INPUT | 要读取的文件 |
| allocated_halo_depth | HOST | INPUT | 用于 halo 交换的深度，现在必须等于 num_import_rings |
| num_partitions | HOST | INPUT | 分区数 |
| partition_sizes | HOST | INPUT | 大小为 num_partitions 的数组，存储每个分区的大小。如果为 NULL，则会从分区向量中计算得到 |
| partition_vector_size | HOST | INPUT | partition_vector 的长度 |
| partition_vector | HOST | INPUT | 全局矩阵的分区分配 |

**输出**

| 参数 | 内存 | 输入/输出 | 含义 |
|---|---|---|---|
| n | HOST | OUTPUT | 局部矩阵的维度 |
| nnz | HOST | OUTPUT | 非零块的数量 |
| block_dimx | HOST | OUTPUT | 块的 x 维度 |
| block_dimy | HOST | OUTPUT | 块的 y 维度 |
| row_ptrs | HOST | OUTPUT | 指向全局矩阵中所有局部矩阵索引的指针 |
| col_indices | HOST | OUTPUT | 指向全局矩阵中所有局部矩阵列索引的指针 |
| data | HOST | OUTPUT | 指向所有局部矩阵值的指针 |
| diag_data | HOST | OUTPUT | 指向所有局部矩阵每行的外部对角线元素的指针如果对角线数据包含在矩阵数据中，则设置为 NULL |
| rhs | HOST | OUTPUT | 指向右侧向量的指针 |
| sol | HOST | OUTPUT | 指向解向量的指针 |
| num_neighbors | HOST | OUTPUT | 将通过边界交换与局部进程交换数据的 MPI 排名的数量 |
| neighbors | HOST | OUTPUT | 指针，指向与此进程共享进程边界的每个 MPI 进程的索引 |
| send_sizes | HOST | OUTPUT | 指向一个包含将发送给相邻进程的每个局部行的数量的数组 |
| send_maps | HOST | OUTPUT | 指向指针数组，其中第 i 个数组包含将发送给相邻进程的局部矩阵的行索引 |
| recv_sizes | HOST | OUTPUT | 指向一个包含将从相邻进程接收的每个相邻节点的局部行数的数组 |
| recv_maps | HOST | OUTPUT | 指向指针数组，其中第 i 个数组包含将从相邻进程接收的相邻节点的局部矩阵的行索引 |

### 2.2.17 MCSOLVERIT_free_system_maps_one_ring()

```
MCSOLVERIT_RC MCSOLVERIT_API MCSOLVERIT_free_system_maps_one_ring(
                                int *row_ptrs,
                                int *col_indices,
                                void *data,
                                void *diag_data,
                                void *rhs,
                                void *sol,
                                int num_neighbors,
                                int *neighbors,
                                int *send_sizes,
                                int **send_maps,
                                int *recv_sizes,
                                int **recv_maps)
```

由 MCSOLVERIT_read_system_maps_one_ring 分配的空闲缓冲区。

**输入**

| 参数 | 内存 | 输入/输出 | 含义 |
|---|---|---|---|
| row_ptrs | HOST | INPUT | 全局矩阵中所有局部矩阵的索引 |
| col_indices | HOST | INPUT | 全局矩阵中所有局部矩阵的列索引 |
| data | HOST | INPUT | 所有局部矩阵的值 |
| diag_data | HOST | INPUT | 所有局部矩阵的每一行的外部对角线元素。如果对角线数据包含在矩阵数据中，则将其设置为 NULL |
| rhs | HOST | INPUT | 右侧向量 |
| sol | HOST | INPUT | 解向量 |
| num_neighbors | HOST | INPUT | 通过边界数据交换进行数据交换的 MPI 排名 |
| neighbors | HOST | INPUT | 与该进程共享边界的每个 MPI 进程的索引 |
| send_sizes | HOST | INPUT | 发送给相邻节点的每个局部行的数量 |
| send_maps | HOST | INPUT | 第 i 个数组包含将发送给相邻节点的局部矩阵的行索引 |
| recv_sizes | HOST | INPUT | 从相邻节点接收到的相邻的局部行数的数量 |
| recv_maps | HOST | INPUT | 第 i 个数组包含了将要从相邻节点接收的相邻节点的局部矩阵中的行索引值 |

### 2.2.18 MCSOLVERIT_write_system()

```
MCSOLVERIT_RC MCSOLVERIT_API MCSOLVERIT_write_system(
                                const MCSOLVERIT_matrix_handle mtx,
                                const MCSOLVERIT_vector_handle rhs,
                                const MCSOLVERIT_vector_handle sol,
                                const char *filename)
```

写一个线性系统到“.mtx”文件中。

**输入**

| 参数 | 内存 | 输入/输出 | 含义 |
|---|---|---|---|
| mtx | HOST | INPUT | 系数矩阵的句柄 |
| rhs | HOST | INPUT | 右侧向量的句柄 |
| sol | HOST | INPUT | 解向量的句柄 |
| filename | HOST | INPUT | 要写入的文件 |

### 2.2.19 MCSOLVERIT_write_system_distributed()

```
MCSOLVERIT_RC MCSOLVERIT_API MCSOLVERIT_write_system_distributed(
                                const MCSOLVERIT_matrix_handle mtx,
                                const MCSOLVERIT_vector_handle rhs,
                                const MCSOLVERIT_vector_handle sol,
                                const char *filename,
                                int allocated_halo_depth,
                                int num_partitions,
                                const int *partition_sizes,
                                int partition_vector_size,
                                const int *partition_vector)
```

写一个分布式线性系统到“.mtx”文件中。

**输入**

| 参数 | 内存 | 输入/输出 | 含义 |
|---|---|---|---|
| mtx | HOST | INPUT | 系数矩阵的句柄 |
| rhs | HOST | INPUT | 右侧向量的句柄 |
| sol | HOST | INPUT | 解向量的句柄 |
| filename | HOST | INPUT | 要写入的文件 |
| allocated_halo_depth | HOST | INPUT | 用于 halo 交换的深度，现在必须等于 num_import_rings |
| num_partitions | HOST | INPUT | 分区数量 |
| partition_sizes | HOST | INPUT | 大小为 num_partitions 的数组存储了每个分区的大小。如果为空，则根据划分向量进行计算 |
| partition_vector_size | HOST | INPUT | partition_vector 的长度 |
| partition_vector | HOST | INPUT | 全局矩阵的划分分配 |

## 2.3 配置引用

### 2.3.1 MCSOLVERIT_config_create()

```
MCSOLVERIT_RC MCSOLVERIT_API MCSOLVERIT_config_create(
                              MCSOLVERIT_config_handle *cfg,
                              const char *options)
```

从字符串创建配置句柄。

**输入**

| 参数 | 内存 | 输入/输出 | 含义 |
|---|---|---|---|
| options | HOST | INPUT | 正确格式的求解器选项 |

**输出**

| 参数 | 内存 | 输入/输出 | 含义 |
|---|---|---|---|
| cfg | HOST | OUTPUT | 指向配置句柄的指针 |

### 2.3.2 MCSOLVERIT_config_create_from_file()

```
MCSOLVERIT_RC MCSOLVERIT_API MCSOLVERIT_config_create_from_file(
                                MCSOLVERIT_config_handle *cfg,
                                const char *param_file)
```

从文件创建配置句柄。

**输入**

| 参数 | 内存 | 输入/输出 | 含义 |
|---|---|---|---|
| param_file | HOST | INPUT | 文件包含正确格式的求解器选项 |

**输出**

| 参数 | 内存 | 输入/输出 | 含义 |
| --- | --- | --- | --- |
| cfg | HOST | OUTPUT | 指向配置句柄的指针 |

### 2.3.3 MCSOLVERIT_config_get_default_number_of_rings()

```
MCSOLVERIT_RC MCSOLVERIT_API MCSOLVERIT_config_get_default_number_of_rings(
                                MCSOLVERIT_config_handle cfg,
                                int *num_import_rings)
```

获取默认的环数。

**输入**

| 参数 | 内存 | 输入/输出 | 含义 |
| --- | --- | --- | --- |
| cfg | HOST | INPUT | 配置句柄 |

**输出**

| 参数 | 内存 | 输入/输出 | 含义 |
| --- | --- | --- | --- |
| num_import_rings | HOST | OUTPUT | 默认环数 |

### 2.3.4 MCSOLVERIT_config_destroy()

```
MCSOLVERIT_RC MCSOLVERIT_API MCSOLVERIT_config_destroy(
                                MCSOLVERIT_config_handle cfg)
```

销毁配置句柄。

**输入**

| 参数 | 内存 | 输入/输出 | 含义 |
| --- | --- | --- | --- |
| cfg | HOST | INPUT | 配置句柄 |

## 2.4 资源引用

### 2.4.1 MCSOLVERIT_resources_create()

```
MCSOLVERIT_RC MCSOLVERIT_API MCSOLVERIT_resources_create(
                                MCSOLVERIT_resources_handle *rsc,
                                MCSOLVERIT_config_handle cfg,
                                void *comm,
                                int device_num,
                                const int *devices)
```

创建资源句柄。

**输入**

| 参数 | 内存 | 输入/输出 | 含义 |
| --- | --- | --- | --- |
| cfg | HOST | INPUT | 配置句柄 |
| comm | HOST | INPUT | 指向通信规范的指针。在 MPI 中，它是 MPI_Comm *。在单进程中，它是 NULL(0) |
| device_num | HOST | INPUT | 此 MPI 进程使用的 GPU 设备数量。目前仅支持 1 个 |
| devices | HOST | INPUT | 此 MPI 进程使用的 GPU 设备的索引 |

**输出**

| 参数 | 内存 | 输入/输出 | 含义 |
| --- | --- | --- | --- |
| rsc | HOST | OUTPUT | 指向资源句柄的指针 |

### 2.4.2 MCSOLVERIT_resources_create_simple()

```
MCSOLVERIT_RC MCSOLVERIT_API MCSOLVERIT_resources_create_simple(
                                MCSOLVERIT_resources_handle *rsc,
                                MCSOLVERIT_config_handle cfg)
```

在单线程应用程序中创建资源句柄。

**输入**

| 参数 | 内存 | 输入/输出 | 含义 |
| --- | --- | --- | --- |
| cfg | HOST | INPUT | 配置句柄 |

**输出**

| 参数 | 内存 | 输入/输出 | 含义 |
| --- | --- | --- | --- |
| rsc | HOST | OUTPUT | 指向资源句柄的指针 |

### 2.4.3 MCSOLVERIT_resources_destroy()

```
MCSOLVERIT_RC MCSOLVERIT_API MCSOLVERIT_resources_destroy(
                                MCSOLVERIT_resources_handle rsc)
```

销毁资源句柄。

**输入**

| 参数 | 内存 | 输入/输出 | 含义 |
| --- | --- | --- | --- |
| rsc | HOST | INPUT | 资源句柄对象 |

## 2.5 矩阵引用

### 2.5.1 MCSOLVERIT_matrix_create()

```
MCSOLVERIT_RC MCSOLVERIT_API MCSOLVERIT_matrix_create(
                                MCSOLVERIT_matrix_handle *mtx,
                                MCSOLVERIT_resources_handle rsc,
                                MCSOLVERIT_Mode mode)
```

创建矩阵句柄。

**输入**

| 参数 | 内存 | 输入/输出 | 含义 |
|---|---|---|---|
| rsc | HOST | INPUT | 资源句柄对象 |
| mode | HOST | INPUT | 求解器模式 |

**输出**

| 参数 | 内存 | 输入/输出 | 含义 |
|---|---|---|---|
| mtx | HOST | OUTPUT | 指向矩阵句柄的指针 |

### 2.5.2 MCSOLVERIT_matrix_destroy()

```
MCSOLVERIT_RC MCSOLVERIT_API MCSOLVERIT_matrix_destroy(
                                MCSOLVERIT_matrix_handle mtx)
```

销毁矩阵句柄。

**输入**

| 参数 | 内存 | 输入/输出 | 含义 |
|---|---|---|---|
| mtx | HOST | INPUT | 矩阵句柄对象 |

### 2.5.3 MCSOLVERIT_matrix_upload_all()

```
MCSOLVERIT_RC MCSOLVERIT_API MCSOLVERIT_matrix_upload_all(
                                MCSOLVERIT_matrix_handle mtx,
                                int n,
                                int nnz,
                                int block_dimx,
                                int block_dimy,
                                const int *row_ptrs,
                                const int *col_indices,
                                const void *data,
                                const void *diag_data)
```

将矩阵上传到矩阵句柄。

**输入**

| 参数 | 内存 | 输入/输出 | 含义 |
|---|---|---|---|
| mtx | HOST | INPUT | 矩阵句柄 |
| n | HOST | INPUT | 每个块中局部矩阵的维度 |
| nnz | HOST | INPUT | 非零块的数量 |
| block_dimx | HOST | INPUT | 块的 x 维度 |
| block_dimy | HOST | INPUT | 块的 y 维度 |
| row_ptrs | HOST/DEVICE | INPUT | 全局矩阵中所有局部矩阵的索引 |
| col_indices | HOST/DEVICE | INPUT | 全局矩阵中所有局部矩阵的列索引 |
| data | HOST/DEVICE | INPUT | 所有局部矩阵的数值按顺序排列 |
| diag_data | HOST/DEVICE | INPUT | 所有局部矩阵的每行外部对角线元素。如果对角线数据包含在矩阵数据中，则将其设置为 NULL |

### 2.5.4 MCSOLVERIT_matrix_upload_all_global()

```
MCSOLVERIT_RC MCSOLVERIT_API MCSOLVERIT_matrix_upload_all_global(
                                MCSOLVERIT_matrix_handle mtx,
                                int n_global,
                                int n,
                                int nnz,
                                int block_dimx,
                                int block_dimy,
                                const int *row_ptrs,
                                const void *col_indices_global,
                                const void *data,
                                const void *diag_data,
                                int allocated_halo_depth,
                                int num_import_rings,
                                const int *partition_vector)
```

将矩阵上传到矩阵句柄。

**输入**

| 参数 | 内存 | 输入/输出 | 含义 |
|---|---|---|---|
| n_global | HOST | INPUT | 每行/列中的块数 |
| n | HOST | INPUT | 每个块中局部矩阵的维度 |
| nnz | HOST | INPUT | 非零块的数量 |
| block_dimx | HOST | INPUT | 块的 x 维度 |
| block_dimy | HOST | INPUT | 块的 y 维度 |
| row_ptrs | HOST | INPUT | 全局矩阵中所有局部矩阵的索引 |
| col_indices_global | HOST | INPUT | 全局矩阵中所有局部矩阵的列索引 |
| data | HOST | INPUT | 所有局部矩阵的数值按顺序排列 |
| diag_data | HOST | INPUT | 所有局部矩阵的每行外部对角线元素。如果对角线数据包含在矩阵数据中，则将其设置为 NULL |
| allocated_halo_depth | HOST | INPUT | 用于 halo 交换的深度，现在必须等于 num_import_rings 的数量 |
| num_import_rings | HOST | INPUT | 指定要重叠的环数 |
| partition_vector | HOST | INPUT | 全局矩阵的分区分配 |

**输出**

| 参数 | 内存 | 输入/输出 | 含义 |
|---|---|---|---|
| mtx | HOST | OUTPUT | 矩阵句柄 |

### 2.5.5 MCSOLVERIT_matrix_replace_coefficients()

```
MCSOLVERIT_RC MCSOLVERIT_API MCSOLVERIT_matrix_replace_coefficients(
                                MCSOLVERIT_matrix_handle mtx,
                                int n,
                                int nnz,
                                const void *data,
                                const void *diag_data)
```

替换矩阵句柄中的矩阵数据。

**输入**

| 参数 | 内存 | 输入/输出 | 含义 |
|---|---|---|---|
| mtx | HOST | INPUT | 矩阵句柄 |
| n | HOST | INPUT | 每个块中局部矩阵的维度 |
| nnz | HOST | INPUT | 非零块的数量 |
| data | HOST/DEVICE | INPUT | 序列中所有局部矩阵的值 |
| diag_data | HOST/DEVICE | INPUT | 所有局部矩阵的每行外部对角线元素<br>如果对角线数据包含在矩阵数据中，则将其设置为 NULL |

### 2.5.6 MCSOLVERIT_matrix_get_size()

```
MCSOLVERIT_RC MCSOLVERIT_API MCSOLVERIT_matrix_get_size(
                                const MCSOLVERIT_matrix_handle mtx,
                                int *n,
                                int *block_dimx,
                                int *block_dimy)
```

得到矩阵的大小

**输入**

| 参数 | 内存 | 输入/输出 | 含义 |
|---|---|---|---|
| mtx | HOST | INPUT | 矩阵句柄 |

**输出**

| 参数 | 内存 | 输入/输出 | 含义 |
|---|---|---|---|
| n | HOST | OUTPUT | 指向每个块中局部矩阵维度的指针 |
| block_dimx | HOST | OUTPUT | 指向块的 x 维度的指针 |
| block_dimy | HOST | OUTPUT | 指向块的 y 维度的指针 |

### 2.5.7 MCSOLVERIT_matrix_comm_from_maps()

```
MCSOLVERIT_RC MCSOLVERIT_API MCSOLVERIT_matrix_comm_from_maps(
                                MCSOLVERIT_matrix_handle mtx,
                                int allocated_halo_depth,
                                int num_import_rings,
                                int max_num_neighbors,
                                const int *neighbors,
                                const int *send_ptrs,
```

(下页继续)

(续上页)

```
                                const int *send_maps,
                                const int *recv_ptrs,
                                const int *recv_maps)
```

创建通信映射。

**输入**

| 参数 | 内存 | 输入/输出 | 含义 |
|---|---|---|---|
| mtx | HOST | INPUT | 矩阵句柄 |
| allocated_halo_depth | HOST | INPUT | 用于 halo 交换的深度，现在必须等于 num_import_rings 的数量 |
| num_import_rings | HOST | INPUT | 指定要重叠的环数 |
| max_num_neighbors | HOST | INPUT | 单个环可以引用的相邻节点数量 |
| neighbors | HOST | INPUT | 与该进程共享边界的每个 MPI 进程的索引 |
| send_ptrs | HOST | INPUT | 在 send_maps 中，环 i 的起始索引 [i * max_num_neighbors + j] 对应于相邻节点 neighbors[j] |
| send_maps | HOST | INPUT | 第 i 个数组包含将发送给相邻节点的局部矩阵的行索引 |
| recv_ptrs | HOST | INPUT | 与 send_ptrs 类似，唯一不同的是将 send_maps 替换为 recv_maps |
| recv_maps | HOST | INPUT | 第 i 个数组包含从相邻节点接收的相邻节点的局部矩阵的行索引 |

### 2.5.8 MCSOLVERIT_matrix_comm_from_maps_one_ring()

```
MCSOLVERIT_RC MCSOLVERIT_API MCSOLVERIT_matrix_comm_from_maps_one_ring(
                                MCSOLVERIT_matrix_handle mtx,
                                int allocated_halo_depth,
                                int num_neighbors,
                                const int *neighbors,
                                const int *send_sizes,
                                const int **send_maps,
                                const int *recv_sizes,
                                const int **recv_maps)
```

创建具有一层重叠的通信映射。

**输入**

| 参数 | 内存 | 输入/输出 | 含义 |
| --- | --- | --- | --- |
| mtx | HOST | INPUT | 矩阵句柄 |
| allocated_halo_depth | HOST | INPUT | 用于 halo 交换的深度，现在必须等于 num_import_rings 的数量 |
| num_neighbors | HOST | INPUT | 将通过边界交换方式进行数据交换的 MPI 排名数量 |
| neighbors | HOST | INPUT | 与该进程共享边界的每个 MPI 进程的索引 |
| send_sizes | HOST | INPUT | 发送给相邻节点的每个局部行数 |
| send_maps | HOST | INPUT | 第 i 个数组包含将发送给相邻节点的局部矩阵的行索引 |
| recv_sizes | HOST | INPUT | 从相邻节点接收的相邻节点局部行数 |
| recv_maps | HOST | INPUT | 第 i 个数组包含从相邻节点接收的局部矩阵的行索引 |

## 2.6 向量的参考

### 2.6.1 MCSOLVERIT_vector_create()

```
MCSOLVERIT_RC MCSOLVERIT_API MCSOLVERIT_vector_create(
                              MCSOLVERIT_vector_handle *vec,
                              MCSOLVERIT_resources_handle rsc,
                              MCSOLVERIT_Mode mode)
```

创建向量句柄。

**输入**

| 参数 | 内存 | 输入/输出 | 含义 |
| --- | --- | --- | --- |
| rsc | HOST | INPUT | 资源句柄 |
| mode | HOST | INPUT | 求解器模式 |

**输出**

| 参数 | 内存 | 输入/输出 | 含义 |
| --- | --- | --- | --- |
| vec | HOST | OUTPUT | 指向向量句柄的指针 |

### 2.6.2 MCSOLVERIT_vector_destroy()

```
MCSOLVERIT_RC MCSOLVERIT_API MCSOLVERIT_vector_destroy(
                                 MCSOLVERIT_vector_handle vec)
```

销毁向量句柄。

**输入**

| 参数 | 内存 | 输入/输出 | 含义 |
| --- | --- | --- | --- |
| vec | HOST | INPUT | 向量句柄 |

### 2.6.3 MCSOLVERIT_vector_upload()

```
MCSOLVERIT_RC MCSOLVERIT_API MCSOLVERIT_vector_upload(
                                MCSOLVERIT_vector_handle vec,
                                int n,
                                int block_dim,
                                const void *data)
```

上传向量到向量句柄。

**输入**

| 参数 | 内存 | 输入/输出 | 含义 |
|---|---|---|---|
| vec | HOST | INPUT | 向量句柄 |
| n | HOST | INPUT | 要上传的向量长度 |
| block_dim | HOST | INPUT | 块的大小 |
| data | HOST/DEVICE | INPUT | 即将上传的向量数据 |

### 2.6.4 MCSOLVERIT_vector_download()

```
MCSOLVERIT_RC MCSOLVERIT_API MCSOLVERIT_vector_download(
                                const MCSOLVERIT_vector_handle vec,
                                void *data)
```

从向量句柄下载向量数据。

**输入**

| 参数 | 内存 | 输入/输出 | 含义 |
|---|---|---|---|
| vec | HOST | INPUT | 向量句柄 |

**输出**

| 参数 | 内存 | 输入/输出 | 含义 |
|---|---|---|---|
| data | HOST/DEVICE | OUTPUT | 下载向量数据 |

### 2.6.5 MCSOLVERIT_vector_set_zero()

```
MCSOLVERIT_RC MCSOLVERIT_API MCSOLVERIT_vector_set_zero(
                                MCSOLVERIT_vector_handle vec,
                                int n,
                                int block_dim)
```

分配并设置向量的值为 0。

**输入**

| 参数 | 内存 | 输入/输出 | 含义 |
|---|---|---|---|
| vec | HOST | INPUT | 向量句柄 |
| n | HOST | INPUT | 每个块中向量数据数 |
| block_dim | HOST | INPUT | 块的大小 |

### 2.6.6 MCSOLVERIT_vector_get_size()

```
MCSOLVERIT_RC MCSOLVERIT_API MCSOLVERIT_vector_get_size(
                                const MCSOLVERIT_vector_handle vec,
                                int *n,
                                int *block_dim)
```

得到向量的大小。

**输入**

| 参数 | 内存 | 输入/输出 | 含义 |
|---|---|---|---|
| vec | HOST | INPUT | 向量句柄 |

**输出**

| 参数 | 内存 | 输入/输出 | 含义 |
|---|---|---|---|
| n | HOST | OUTPUT | 指向每个块中向量数据的数目 |
| block_dim | HOST | OUTPUT | 指向块的维度的指针 |

### 2.6.7 MCSOLVERIT_vector_bind()

```
MCSOLVERIT_RC MCSOLVERIT_API MCSOLVERIT_vector_bind(
                                MCSOLVERIT_vector_handle vec,
                                const MCSOLVERIT_matrix_handle mtx)
```

通过从矩阵中复制数据，创建向量的通信映射和分区信息。

**输入**

| 参数 | 内存 | 输入/输出 | 含义 |
|---|---|---|---|
| vec | HOST | INPUT | 向量句柄 |
| mtx | HOST | INPUT | 矩阵句柄 |

## 2.7 求解器参考

### 2.7.1 MCSOLVERIT_solver_create()

```
MCSOLVERIT_RC MCSOLVERIT_API MCSOLVERIT_solver_create(
                                MCSOLVERIT_solver_handle *slv,
                                MCSOLVERIT_resources_handle rsc,
                                MCSOLVERIT_Mode mode,
                                const MCSOLVERIT_config_handle cfg_solver)
```

创建求解器句柄.

**输入**

| 参数名 | 存储 | 输入/输出 | 含义 |
|---|---|---|---|
| rsc \| HOST | | INPUT | 资源句柄 |
| mode \| HOST | | INPUT | 求解器模式 |
| cfg_solver \| HOST | | INPUT | 配置句柄 |

**输出**

| 参数 | 内存 | 输入/输出 | 含义 |
| --- | --- | --- | --- |
| slv | HOST | OUTPUT | 指向求解器句柄的指针 |

### 2.7.2 MCSOLVERIT_solver_destroy()

```
MCSOLVERIT_RC MCSOLVERIT_API MCSOLVERIT_solver_destroy(
                                MCSOLVERIT_solver_handle slv)
```

销毁求解器句柄。

**输入**

| 参数 | 内存 | 输入/输出 | 含义 |
| --- | --- | --- | --- |
| slv | HOST | INPUT | 求解器句柄 |

### 2.7.3 MCSOLVERIT_solver_setup()

```
MCSOLVERIT_RC MCSOLVERIT_API MCSOLVERIT_solver_setup(
                                MCSOLVERIT_solver_handle slv,
                                MCSOLVERIT_matrix_handle mtx)
```

求解器设置阶段。

**输入**

| 参数 | 内存 | 输入/输出 | 含义 |
| --- | --- | --- | --- |
| slv | HOST | INPUT | 求解器句柄 |
| mtx | HOST | INPUT | 系数矩阵句柄 |

### 2.7.4 MCSOLVERIT_solver_solve()

```
MCSOLVERIT_RC MCSOLVERIT_API MCSOLVERIT_solver_solve(
                                MCSOLVERIT_solver_handle slv,
                                MCSOLVERIT_vector_handle rhs,
                                MCSOLVERIT_vector_handle sol)
```

求解器求解阶段。

**输入**

| 参数 | 内存 | 输入/输出 | 意义 |
| --- | --- | --- | --- |
| slv | HOST | INPUT | 求解器句柄 |
| rhs | HOST | INPUT | 右侧向量句柄 |

**输出**

| 参数 | 内存 | 输入/输出 | 意义 |
| --- | --- | --- | --- |
| sol | HOST | OUTPUT | 解向量句柄 |

### 2.7.5 MCSOLVERIT_solver_solve_with_0_initial_guess()

```
MCSOLVERIT_RC MCSOLVERIT_API MCSOLVERIT_solver_solve_with_0_initial_guess(
                                MCSOLVERIT_solver_handle slv,
                                MCSOLVERIT_vector_handle rhs,
                                MCSOLVERIT_vector_handle sol)
```

求解器求解阶段，初始值猜测为 0。

**输入**

| 参数 | 内存 | 输入/输出 | 含义 |
|---|---|---|---|
| slv | HOST | INPUT | 求解器句柄 |
| rhs | HOST | INPUT | 右侧向量句柄 |

**输出**

| 参数 | 内存 | 输入/输出 | 含义 |
|---|---|---|---|
| sol | HOST | OUTPUT | 解向量句柄 |

### 2.7.6 MCSOLVERIT_solver_get_iterations_number()

```
MCSOLVERIT_RC MCSOLVERIT_API MCSOLVERIT_solver_get_iterations_number(
                                MCSOLVERIT_solver_handle slv,
                                int *n)
```

得到求解阶段的精确迭代次数。

**输入**

| 参数 | 内存 | 输入/输出 | 含义 |
|---|---|---|---|
| slv | HOST | INPUT | 求解器句柄 |

**输出**

| 参数 | 内存 | 输入/输出 | 含义 |
|---|---|---|---|
| n | HOST | OUTPUT | 指向迭代次数的指针 |

### 2.7.7 MCSOLVERIT_solver_get_iteration_residual()

```
MCSOLVERIT_RC MCSOLVERIT_API MCSOLVERIT_solver_get_iteration_residual(
                                MCSOLVERIT_solver_handle slv,
                                int it,
                                int idx,
                                double *res)
```

获取给定迭代 `it` 和块索引 `idx` 的残差。

**输入**

| 参数 | 内存 | 输入/输出 | 含义 |
|---|---|---|---|
| slv | HOST | INPUT | 求解器句柄 |
| it | HOST | INPUT | 获取残差的迭代次数 |
| idx | HOST | INPUT | 获取残差的块索引 |

**输出**

| 参数 | 内存 | 输入/输出 | 含义 |
| --- | --- | --- | --- |
| res | HOST | OUTPUT | 指向残差的指针 |

### 2.7.8 MCSOLVERIT_solver_get_status()

```
MCSOLVERIT_RC MCSOLVERIT_API MCSOLVERIT_solver_get_status(
                                MCSOLVERIT_solver_handle slv,
                                MCSOLVERIT_SOLVE_STATUS *st)
```

获取解决阶段的状态。

**输入**

| 参数 | 内存 | 输入/输出 | 含义 |
| --- | --- | --- | --- |
| slv | HOST | INPUT | 求解器句柄 |

**输出**

| 参数 | 内存 | 输入/输出 | 含义 |
| --- | --- | --- | --- |
| st | HOST | INPUT | 指向求解状态的指针 |