PSkel

http://pskel.github.io

Generated by Doxygen 1.7.6.1

Fri Jun 12 2015 02:17:18

Contents

Chapter 1

PSkel: High-performance parallel skeletons

1.1 Introduction

PSkel is a high-performance framework for parallel skeletons. Using a high-level abstraction for parallel skeletons, PSkel releases the programmer from the responsibility of writing boiler-plate code for parallel programming in heterogeneous architectures, e.g., explicit synchronization and data exchanges between GPU memory and main memory. Furthermore, the framework translates the abstractions described using its application programming interface (API) into lowlevel C++ code compatible with Intel TBB, OpenMP and NVIDIA CUDA. PSkel's API is mainly based on a C++ template library that implements parallel skeletons and provides useful constructs for developing parallel applications. The framework provides an API for manipulating input and output data; specifying stencil computations; encapsulating memory management, computations, and runtime details.

Chapter 2

Class Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 5

Class Documentation

5.1 PSkel::Args< T > Class Template Reference

Public Member Functions

- **Args** (int _width)
- __device__ __host__ int **getWidth** () const
- __device__ __host__ T & **operator()** (int x) const

Public Attributes

- T * **hostArray**
- T * **deviceArray**
- int **width**

template<typename T> class PSkel::Args< T >

The documentation for this class was generated from the following files:

- src/PSkelArgs.h
- src/PSkelArgs.hpp

5.2 PSkel::Args2D< T > Class Template Reference

Public Member Functions

- **Args2D** (int _width, int _height)
- __device__ __host__ int **getWidth** () const
- __device__ __host__ int **getHeight** () const
- __device__ __host__ T & **operator()** (int x, int y) const

Public Attributes

- T * **hostArray**
- T * **deviceArray**
- int **width**
- int **height**

template<typename T> class PSkel::Args2D< T >

The documentation for this class was generated from the following files:

- src/PSkelArgs.h
- src/PSkelArgs.hpp

5.3 PSkel::Args3D< T > Class Template Reference

Public Member Functions

- **Args3D** (int _width, int _height, int _depth)
- __device__ __host__ int **getWidth** () const
- __device__ __host__ int **getHeight** () const
- __device__ __host__ int **getDepth** () const
- __device__ __host__ T & **operator()** (int x, int y, int z) const

Public Attributes

- T * **hostArray**
- T * **deviceArray**
- int **width**
- int **height**
- int **depth**

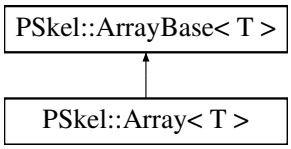
template<typename T> class PSkel::Args3D< T >

The documentation for this class was generated from the following files:

- src/PSkelArgs.h
- src/PSkelArgs.hpp

5.4 PSkel::Array< T > Class Template Reference

Inheritance diagram for PSkel::Array< T >:



Public Member Functions

- Array ()
- Array (size_t size)
- __attribute__ ((always_inline)) __forceinline__ __device__ __host__ T &operator()(size_t w) const

template<typename T> class PSkel::Array< T >

5.4.1 Constructor & Destructor Documentation

5.4.1.1 template<typename T > PSkel::Array< T >::Array ( )

The Array default constructor creates an empty array withour allocating memory space.

5.4.1.2 template<typename T > PSkel::Array< T >::Array (size_t *size*)

The Array constructor creates and allocates the specified 1-dimensional array in the host memory.

Parameters

in	*size*	size for the 1D array being created.

5.4.2 Member Function Documentation

5.4.2.1 template<typename T> PSkel::Array< T >::__attribute__ ((always_inline)) const

Access a specific element of the array allocated in the memory space relative to the execution environment, i.e. either in the host or device memory.

Parameters

in	*w*	offset for the element being accessed.

Returns

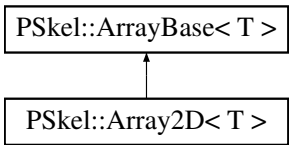
the reference of the element specified via parameters.

The documentation for this class was generated from the following files:

- src/PSkelArray.h
- src/PSkelArray.hpp

5.5 PSkel::Array2D< T > Class Template Reference

Inheritance diagram for PSkel::Array2D< T >:



Public Member Functions

- Array2D ()
- Array2D (size_t width, size_t height)
- __attribute__ ((always_inline)) __forceinline__ __device__ __host__ T &operator()(size_t h

Public Attributes

- size_t w **const**

template<typename T> class PSkel::Array2D< T >

5.5.1 Constructor & Destructor Documentation

5.5.1.1 template<typename T > PSkel::Array2D< T >::Array2D ( )

The Array2D default constructor creates an empty array withour allocating memory space.

5.5.1.2 template<typename T > PSkel::Array2D< T >::Array2D (size_t *width*, size_t *height*)

The Array2D constructor creates and allocates the specified 2-dimensional array in the host memory.

Parameters

in	*width*	width for the 2D array being created.
in	*height*	height for the 2D array being created.

5.5.2 Member Function Documentation

5.5.2.1 template<typename T> PSkel::Array2D< T >::__attribute__ ((always_inline))

Access a specific element of the array allocated in the memory space relative to the execution environment, i.e. either in the host or device memory.

Parameters

in	*h*	height offset for the element being accessed.
in	*w*	width offset for the element being accessed.

Returns

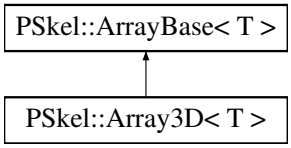
the reference of the element specified via parameters.

The documentation for this class was generated from the following files:

- src/PSkelArray.h
- src/PSkelArray.hpp

5.6 PSkel::Array3D< T > Class Template Reference

Inheritance diagram for PSkel::Array3D< T >:



Public Member Functions

- Array3D ()

- Array3D (size_t width, size_t height, size_t depth)
- __attribute__ ((always_inline)) __forceinline__ __device__ __host__ T &operator()(size_t h

Public Attributes

- size_t **w**
- size_t size_t d **const**

template<typename T> class PSkel::Array3D< T >

5.6.1 Constructor & Destructor Documentation

5.6.1.1 template<typename T > PSkel::Array3D< T >::Array3D ( )

The Array3D default constructor creates an empty array withour allocating memory space.

5.6.1.2 template<typename T > PSkel::Array3D< T >::Array3D (size_t *width*, size_t *height*, size_t *depth*)

The Array3D constructor creates and allocates the specified 3-dimensional array in the host memory.

Parameters

in	*width*	width for the 3D array being created.
in	*height*	height for the 3D array being created.
in	*depth*	depth for the 3D array being created.

5.6.2 Member Function Documentation

5.6.2.1 template<typename T> PSkel::Array3D< T >::__attribute__ ((always_inline))

Access a specific element of the array allocated in the memory space relative to the execution environment, i.e. either in the host or device memory.

Parameters

in	*h*	height offset for the element being accessed.
in	*w*	width offset for the element being accessed.
in	*d*	depth offset for the element being accessed.

Returns

the reference of the element specified via parameters.

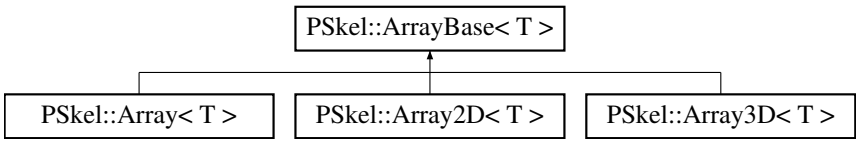
The documentation for this class was generated from the following files:

- src/PSkelArray.h
- src/PSkelArray.hpp

5.7 PSkel::ArrayBase< T > Class Template Reference

`#include <PSkelArray.h>`

Inheritance diagram for PSkel::ArrayBase< T >:



Public Member Functions

- void deviceAlloc ()
- void deviceFree ()
- void **hostAlloc** (size_t width, size_t height, size_t depth)
- void hostAlloc ()
- void hostFree ()
- __device__ __host__ size_t getWidth () const
- __device__ __host__ size_t getHeight () const
- __device__ __host__ size_t getDepth () const
- __device__ __host__ size_t memSize () const
- __device__ __host__ size_t size () const
- __device__ __host__ size_t realSize () const
- template<typename Arrays >
  void hostSlice (Arrays array, size_t widthOffset, size_t heightOffset, size_t depthOffset, size_t width, size_t height, size_t depth)
- template<typename Arrays >
  void hostClone (Arrays array)
- template<typename Arrays >
  void hostMemCopy (Arrays array)
- void copyToDevice ()
- template<typename Arrays >
  void copyFromDevice (Arrays array)
- void copyToHost ()
- __device__ __host__ operator bool () const

Protected Member Functions

- __device__ __forceinline__ T & deviceGet (size_t h, size_t w, size_t d) const
- __host__ __forceinline__ T & hostGet (size_t h, size_t w, size_t d) const
- ArrayBase (size_t width, size_t height, size_t depth)

5.7.1 Detailed Description

template<typename T>class PSkel::ArrayBase< T >

Class that implements the basic data structure used by the parallel skeletons (such as stencil and map.) PSkel::ArrayBase is a 3D array that is also interfaced via 1D and 2D arrays. The PSkel::ArrayBase data structure that is extended by PSkel::Array, PSkel::Array2D, and PSkel::Array3D.

5.7.2 Constructor & Destructor Documentation

5.7.2.1 template<typename T > PSkel::ArrayBase< T >::ArrayBase (size_t *width*, size_t *height*, size_t *depth*) [protected]

The ArrayBase constructor creates and allocates the specified array in the host memory.

Parameters

in	*width*	Width for the 3D array being created.
in	*height*	Height for the 3D array being created.
in	*depth*	Depth for the 3D array being created.

5.7.3 Member Function Documentation

5.7.3.1 template<typename T > template<typename Arrays > void PSkel::ArrayBase< T >::copyFromDevice (Arrays *array*)

The array given as argument is copied from the device allocated memory to the host allocated memory of this array. The data is efficiently transferred from device to host.

Parameters

in	*array*	the source array that holds the data that will be copied from device to the host memory of this array.

5.7.3.2 template<typename T > void PSkel::ArrayBase< T >::copyToDevice ( )

The array is copied from the host allocated memory to the device allocated memory. The data is efficiently transferred from host to device. Both the host and device memory must be allocated before the data is transferred.

5.7.3.3 template<typename T > void PSkel::ArrayBase< T >::copyToHost ( )

The array is copied from the device allocated memory to the host allocated memory. The data is efficiently transferred from device to host. Both the host and device memory must be allocated before the data is transferred.

5.7.3.4 template<typename T > void PSkel::ArrayBase< T >::deviceAlloc ( )

Allocates the "virtual" array in device memory.

5.7.3.5 template<typename T > void PSkel::ArrayBase< T >::deviceFree ( )

Frees the allocated device memory.

5.7.3.6 template<typename T > __device__ __forceinline__ T & PSkel::ArrayBase< T >::deviceGet (size_t *h*, size_t *w*, size_t *d*) const [protected]

Access a specific element of the array allocated in the device memory. This function is accessible only during the execution in the device environment.

Parameters

in	*h*	height offset for the element being accessed.
in	*w*	width offset for the element being accessed.
in	*d*	depth offset for the element being accessed.

Returns

the reference of the element specified via parameters.

5.7.3.7 template<typename T > size_t PSkel::ArrayBase< T >::getDepth ( ) const

Get the depth size of the "virtual" array.

Returns

the "virtual" depth of the array data structure.

5.7.3.8 template<typename T > size_t PSkel::ArrayBase< T >::getHeight ( ) const

Get the height size of the "virtual" array.

Returns

the "virtual" height of the array data structure.

5.7.3.9 template<typename T > size_t PSkel::ArrayBase< T >::getWidth ( ) const

Get the width size of the "virtual" array.

Returns

the "virtual" width of the array data structure.

5.7.3.10 template<typename T > void PSkel::ArrayBase< T >::hostAlloc ( )

Allocates the array in host (main) memory.

5.7.3.11 template<typename T > template<typename Arrays > void PSkel::ArrayBase< T >::hostClone (Arrays *array*)

Creates a clone, in the host (main) memory, of the array given as argument. The clone is a copy of the array in a different memory space.

Parameters

in	*array*	original array that will be cloned.

5.7.3.12 template<typename T > void PSkel::ArrayBase< T >::hostFree ( )

Frees the allocated host (main) memory.

5.7.3.13 template<typename T > T & PSkel::ArrayBase< T >::hostGet (size_t *h*, size_t *w*, size_t *d*) const [protected]

Access a specific element of the array allocated in the host memory. This function is accessible only during the execution in the host environment.

Parameters

in	*h*	height offset for the element being accessed.
in	*w*	width offset for the element being accessed.
in	*d*	depth offset for the element being accessed.

Returns

the reference of the element specified via parameters.

5.7.3.14 template<typename T > template<typename Arrays > void PSkel::ArrayBase< T >::hostMemCopy (Arrays *array*)

Copies the data, in the host (main) memory, from the array given as argument.

Parameters

in	*array*	original array that will be copied.

5.7.3.15 template<typename T > template<typename Arrays > void PSkel::ArrayBase< T >::hostSlice (Arrays *array,* size_t *widthOffset,* size_t *heightOffset,* size_t *depthOffset,* size_t *width,* size_t *height,* size_t *depth*)

Creates a sliced reference, in the host (main) memory, of the array given as argument. The slice points to the same memory space as the sliced array.

Parameters

in	*array*	original array that will be sliced.
in	*widthOffset*	the width offset for the sliced region, relative to the array given as argument.
in	*heightOffset*	the height offset for the sliced region, relative to the array given as argument.
in	*depthOffset*	the depth offset for the sliced region, relative to the array given as argument.
in	*width*	the width of the slice.
in	*height*	the height of the slice.
in	*depth*	the depth of the slice.

5.7.3.16 template<typename T > size_t PSkel::ArrayBase< T >::memSize ( ) const

Get the size, in bytes, of the allocated memory for the "virtual" array.

Returns

the total of bytes allocated in memory for the "virtual" array.

5.7.3.17 template<typename T > PSkel::ArrayBase< T >::operator bool ( ) const

Verifies if there is memory allocated for the array data structure. This function can be called both from device and host environment, and the respective memory space is verified.

Returns

true if there is a valid memory spaced allocated for the array; false otherwise.

5.7.3.18 template<typename T > size_t PSkel::ArrayBase< T >::realSize ( ) const

Get the size of the real allocated array, i.e. the number of elements

Returns

the size of the real allocated array.

5.7.3.19 template<typename T > size_t PSkel::ArrayBase< T >::size ( ) const

Get the size of the "virtual" array, i.e. the number of elements

Returns

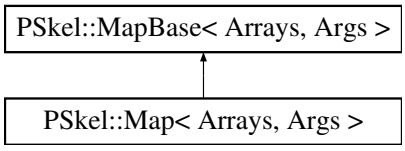
the size of the "virtual" array.

The documentation for this class was generated from the following files:

- src/PSkelArray.h
- src/PSkelArray.hpp

5.8 PSkel::Map< Arrays, Args > Class Template Reference

Inheritance diagram for PSkel::Map< Arrays, Args >:



Public Member Functions

- **Map** (Arrays input, Arrays output, Args args)

Protected Member Functions

- void **runSeq** (Arrays in, Arrays out)
- void **runOpenMP** (size_t numThreads)
- void **runCUDA** (Arrays in, Arrays out, int blockSize)

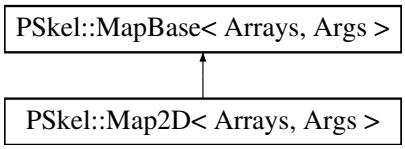
template<class Arrays, class Args> class PSkel::Map< Arrays, Args >

The documentation for this class was generated from the following files:

- src/PSkelMap.h
- src/PSkelMap.hpp

5.9 PSkel::Map2D< Arrays, Args > Class Template Reference

Inheritance diagram for PSkel::Map2D< Arrays, Args >:



Public Member Functions

- **Map2D** (Arrays input, Arrays output, Args args)

Protected Member Functions

- void **runSeq** (Arrays in, Arrays out)
- void **runOpenMP** (size_t numThreads)
- void **runCUDA** (Arrays in, Arrays out, int blockSize)

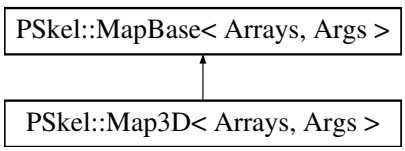
template<class Arrays, class Args> class PSkel::Map2D< Arrays, Args >

The documentation for this class was generated from the following files:

- src/PSkelMap.h
- src/PSkelMap.hpp

5.10 PSkel::Map3D< Arrays, Args > Class Template Reference

Inheritance diagram for PSkel::Map3D< Arrays, Args >:



Public Member Functions

- **Map3D** (Arrays input, Arrays output, Args args)

Protected Member Functions

- void **runSeq** (Arrays in, Arrays out)
- void **runOpenMP** (size_t numThreads)
- void **runCUDA** (Arrays in, Arrays out, int blockSize)

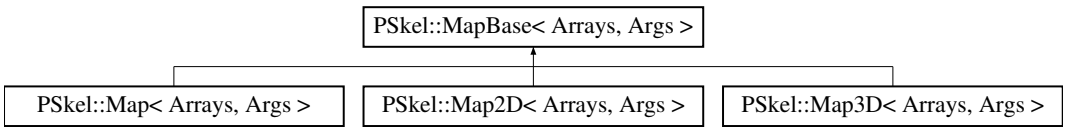
template<class Arrays, class Args> class PSkel::Map3D< Arrays, Args >

The documentation for this class was generated from the following files:

- src/PSkelMap.h
- src/PSkelMap.hpp

5.11 PSkel::MapBase< Arrays, Args > Class Template Reference

Inheritance diagram for PSkel::MapBase< Arrays, Args >:



Public Member Functions

- void **runSequential** ()
- void **runCPU** (size_t numThreads=0)
- void **runGPU** (size_t blockSize=0)
- void **runIterativeSequential** (size_t iterations)
- void **runIterativeCPU** (size_t iterations, size_t numThreads=0)
- void **runIterativeGPU** (size_t iterations, size_t blockSize=0)

Protected Member Functions

- virtual void **runSeq** (Arrays in, Arrays out)=0
- virtual void **runOpenMP** (Array in, Array out, size_t numThreads)=0
- virtual void **runCUDA** (Arrays input, Arrays output, size_t blockSize)=0

Protected Attributes

- Arrays **input**
- Arrays **output**
- Args **args**

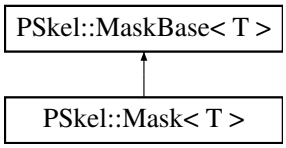
template<class Arrays, class Args = int> class PSkel::MapBase< Arrays, Args >

The documentation for this class was generated from the following files:

- src/PSkelMap.h
- src/PSkelMap.hpp

5.12 PSkel::Mask< T > Class Template Reference

Inheritance diagram for PSkel::Mask< T >:



Public Member Functions

- Mask (size_t size=0, T haloVal=T(0), size_t range=0)
- __device__ __host__ void set (size_t n, int i, T weight=T(0))
- template<typename V >
  __forceinline__ __device__ __host__ T get (size_t n, Array< V > array, size_t i)
- __device__ __host__ size_t **getRange** ()

template<typename T> class PSkel::Mask< T >

5.12.1 Constructor & Destructor Documentation

5.12.1.1 template<typename T > PSkel::Mask< T >::Mask (size_t *size* = 0, T *haloVal* = T(0), size_t *range* = 0)

The Mask constructor creates and allocates the specified 1-dimensional mask in the host memory.

Parameters

in	*size*	the size of the 1D mask.
in	*haloVal*	the value used when the array is accessed out of bounds.
in	*range*	the range of the mask; if range is 0, it is calculated as the maximum absolute value on the mask.

5.12.2 Member Function Documentation

5.12.2.1 template<typename T > template<typename V > T PSkel::Mask< T >::get (size_t *n*, Array< V > *array*, size_t *i*)

Get the n-th neighbor from the specified input array.

Parameters

in	*n*	the index of the neighbor.
in	*array*	the input 1D array.
in	*i*	the index for the central element.

Returns

the n-th neighbor of the given central element, from the input array.

5.12.2.2 template<typename T > void PSkel::Mask< T >::set (size_t *n*, int *i*, T *weight* = `T(0)`)

Set the mask information for accessing the n-th neighbor for a given element.

Parameters

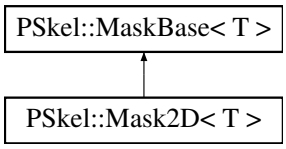
in	*n*	the index of the neighbor.
in	*i*	the index translation needed for acessing the n-th neighbor.
in	*weight*	the weight defined for the n-th neighbor.

The documentation for this class was generated from the following files:

- src/PSkelMask.h
- src/PSkelMask.hpp

5.13 PSkel::Mask2D< T > Class Template Reference

Inheritance diagram for PSkel::Mask2D< T >:



Public Member Functions

- Mask2D (size_t size=0, T haloVal=T(0), size_t range=0)
- __device__ __host__ void set (size_t n, int h, int w, T weight=T(0))
- template<typename V >
  __forceinline__ __device__ __host__ T get (size_t n, Array2D< V > array, size_t h, size_t w)
- __device__ __host__ size_t **getRange** ()

template<typename T> class PSkel::Mask2D< T >

5.13.1 Constructor & Destructor Documentation

5.13.1.1 template<typename T > PSkel::Mask2D< T >::Mask2D (size_t *size* = 0, T *haloVal* = T(0), size_t *range* = 0)

The Mask2D constructor creates and allocates the specified 2-dimensional mask in the host memory.

Parameters

in	*size*	the size of the 2D mask.
in	*haloVal*	the value used when the array is accessed out of bounds.
in	*range*	the range of the mask; if range is 0, it is calculated as the maximum absolute value on the mask.

5.13.2 Member Function Documentation

5.13.2.1 template<typename T > template<typename V > T PSkel::Mask2D< T >::get (size_t *n*, Array2D< V > *array*, size_t *h*, size_t *w*)

Get the n-th neighbor from the specified input array.

Parameters

in	*n*	the index of the neighbor.
in	*array*	the input 2D array.
in	*h*	the height index for the central element.
in	*w*	the width index for the central element.

Returns

the n-th neighbor of the given central element, from the input array.

5.13.2.2 template<typename T > void PSkel::Mask2D< T >::set (size_t *n*, int *h*, int *w*, T *weight* = `T(0)`)

Set the mask information for accessing the n-th neighbor for a given element.

Parameters

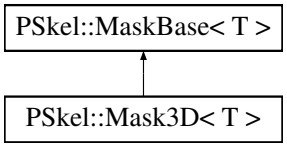
`in`	*n*	the index of the neighbor.
`in`	*h*	the height index translation needed for acessing the n-th neighbor.
`in`	*w*	the width index translation needed for acessing the n-th neighbor.
`in`	*weight*	the weight defined for the n-th neighbor.

The documentation for this class was generated from the following files:

- src/PSkelMask.h
- src/PSkelMask.hpp

5.14 PSkel::Mask3D< T > Class Template Reference

Inheritance diagram for PSkel::Mask3D< T >:



Public Member Functions

- Mask3D (size_t size=0, T haloVal=T(0), size_t range=0)
- __device__ __host__ void set (size_t n, int h, int w, int d, T weight=T(0))
- template<typename V >
  __forceinline__ __device__ __host__ T get (size_t n, Array3D< V > array, size_t h, size_t w, size_t d)
- __device__ __host__ size_t **getRange** ()

template<typename T> class PSkel::Mask3D< T >

5.14.1 Constructor & Destructor Documentation

5.14.1.1 template<typename T > PSkel::Mask3D< T >::Mask3D (size_t *size* = 0, T *haloVal* = T(0), size_t *range* = 0)

The Mask3D constructor creates and allocates the specified 3-dimensional mask in the host memory.

Parameters

in	*size*	the size of the 3D mask.
in	*haloVal*	the value used when the array is accessed out of bounds.
in	*range*	the range of the mask; if range is 0, it is calculated as the maximum absolute value on the mask.

5.14.2 Member Function Documentation

5.14.2.1 template<typename T > template<typename V > T PSkel::Mask3D< T >::get (size_t *n*, Array3D< V > *array*, size_t *h*, size_t *w*, size_t *d*)

Get the n-th neighbor from the specified input array.

Parameters

in	*n*	the index of the neighbor.
in	*array*	the input 3D array.
in	*h*	the height index for the central element.
in	*w*	the width index for the central element.
in	*d*	the depth index for the central element.

Returns

the n-th neighbor of the given central element, from the input array.

5.14.2.2 template<typename T > void PSkel::Mask3D< T >::set (size_t *n*, int *h*, int *w*, int *d*, T *weight* = T(0))

Set the mask information for accessing the n-th neighbor for a given element.

Parameters

in	*n*	the index of the neighbor.
in	*h*	the height index translation needed for acessing the n-th neighbor.
in	*w*	the width index translation needed for acessing the n-th neighbor.
in	*d*	the depth index translation needed for acessing the n-th neighbor.
in	*weight*	the weight defined for the n-th neighbor.

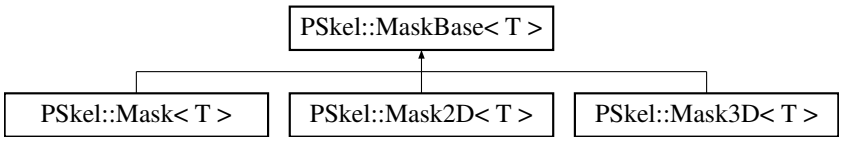
The documentation for this class was generated from the following files:

- src/PSkelMask.h
- src/PSkelMask.hpp

5.15 PSkel::MaskBase< T > Class Template Reference

`#include <PSkelMask.h>`

Inheritance diagram for PSkel::MaskBase< T >:



Public Member Functions

- void deviceAlloc ()
- void **copyToDevice** ()
- void deviceFree ()
- void hostAlloc ()
- void hostFree ()
- size_t memSize () const
- __device__ __host__ T getWeight (size_t n)

Public Attributes

- size_t **size**
- size_t **dimension**
- size_t **range**

Protected Member Functions

- MaskBase (size_t size=0, size_t dimension=0, T haloVal=T(0), size_t range=0)

Protected Attributes

- int * **hostMask**
- int * **deviceMask**
- T * **hostWeight**
- T * **deviceWeight**
- T **haloValue**

5.15.1 Detailed Description

template<typename T>class PSkel::MaskBase< T >

MaskBase is the basic class that implements the mask data structure used by the stencil skeleton in order to select the neighborhood for each element of a given input array.

5.15.2 Constructor & Destructor Documentation

5.15.2.1 template<typename T > PSkel::MaskBase< T >::MaskBase (size_t *size* = 0, size_t *dimension* = 0, T *haloVal* = T(0), size_t *range* = 0) [protected]

The MaskBase constructor creates and allocates the specified mask in the host memory.

Parameters

in	*size*	the size of the mask.
in	*dimension*	the dimension of the mask.
in	*haloVal*	the value used when the array is accessed out of bounds.
in	*range*	the range of the mask; if range is 0, it is calculated as the maximum absolute value on the mask.

5.15.3 Member Function Documentation

5.15.3.1 template<typename T > void PSkel::MaskBase< T >::deviceAlloc ( )

Allocates the mask in device memory, including both the indexes and the weights.

5.15.3.2 template<typename T > void PSkel::MaskBase< T >::deviceFree ( )

Frees the allocated device memory.

5.15.3.3 template<typename T > T PSkel::MaskBase< T >::getWeight (size_t *n*)

Get the weight of a given element in the mask.

Parameters

in	*n*	index of the element in the mask.

Returns

the weight of the specified element.

5.15.3.4 template<typename T > void PSkel::MaskBase< T >::hostAlloc ( )

Allocates the mask in host (main) memory, including both the indexes and the weights.

5.15.3.5 template<typename T > void PSkel::MaskBase< T >::hostFree ( )

Frees the allocated host (main) memory.

5.15.3.6 template<typename T > size_t PSkel::MaskBase< T >::memSize ( ) const

Get the size, in bytes, of the allocated memory for storing the mask.

Returns

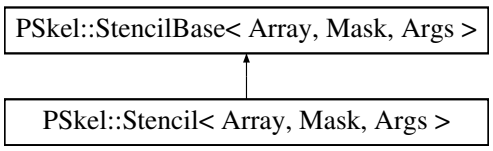
the total of bytes allocated in memory for storing the mask.

The documentation for this class was generated from the following files:

- src/PSkelMask.h
- src/PSkelMask.hpp

5.16 PSkel::Stencil< Array, Mask, Args > Class Template - Reference

Inheritance diagram for PSkel::Stencil< Array, Mask, Args >:



Public Member Functions

- **Stencil** (Array _input, Array _output, Mask _mask, Args _args)

Protected Member Functions

- void **runSeq** (Array in, Array out)
- void **runOpenMP** (Array in, Array out, size_t numThreads)

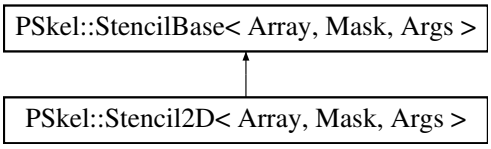
template<class Array, class Mask, class Args> class PSkel::Stencil< Array, Mask, Args >

The documentation for this class was generated from the following files:

- src/PSkelStencil.h
- src/PSkelStencil.hpp

5.17 PSkel::Stencil2D< Array, Mask, Args > Class Template Reference

Inheritance diagram for PSkel::Stencil2D< Array, Mask, Args >:



Public Member Functions

- **Stencil2D** (Array _input, Array _output, Mask _mask, Args _args)

Protected Member Functions

- void **runSeq** (Array in, Array out)
- void **runOpenMP** (Array in, Array out, size_t numThreads)

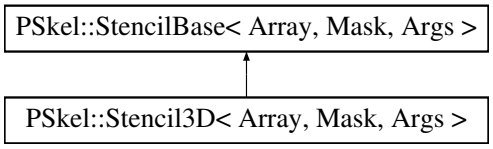
template<class Array, class Mask, class Args> class PSkel::Stencil2D< Array, Mask, Args >

The documentation for this class was generated from the following files:

- src/PSkelStencil.h
- src/PSkelStencil.hpp

5.18 PSkel::Stencil3D< Array, Mask, Args > Class Template Reference

Inheritance diagram for PSkel::Stencil3D< Array, Mask, Args >:



Public Member Functions

- **Stencil3D** (Array _input, Array _output, Mask _mask, Args _args)

Protected Member Functions

- void **runSeq** (Array in, Array out)
- void **runOpenMP** (Array in, Array out, size_t numThreads)

template<class Array, class Mask, class Args> class PSkel::Stencil3D< Array, Mask, Args >

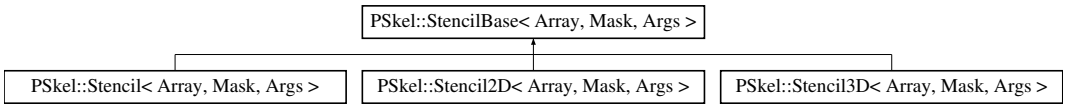
The documentation for this class was generated from the following files:

- src/PSkelStencil.h
- src/PSkelStencil.hpp

5.19 PSkel::StencilBase< Array, Mask, Args > Class Template - Reference

`#include <PSkelStencil.h>`

Inheritance diagram for PSkel::StencilBase< Array, Mask, Args >:



Public Member Functions

- void runSequential ()
- void runCPU (size_t numThreads=0)
- void runGPU (size_t GPUBlockSize=0)
- void runTilingGPU (size_t tilingWidth, size_t tilingHeight, size_t tilingDepth, size_t GPUBlockSize=0)
- void runAutoGPU (size_t GPUBlockSize=0)
- void runIterativeSequential (size_t iterations)

- void runIterativeCPU (size_t iterations, size_t numThreads=0)
- void runIterativeGPU (size_t iterations, size_t GPUBlockSize=0)
- void runIterativeTilingGPU (size_t iterations, size_t tilingWidth, size_t tilingHeight, size_t tilingDepth, size_t innerIterations=1, size_t GPUBlockSize=0)
- void runIterativeAutoGPU (size_t iterations, size_t GPUBlockSize=0)

Protected Member Functions

- virtual void **runSeq** (Array in, Array out)=0
- virtual void **runOpenMP** (Array in, Array out, size_t numThreads)=0
- void **runCUDA** (Array, Array, int)
- void **runIterativeTilingCUDA** (Array in, Array out, StencilTiling< Array, Mask > tiling, size_t GPUBlockSize)

Protected Attributes

- Array **input**
- Array **output**
- Args **args**
- Mask **mask**

5.19.1 Detailed Description

template<class Array, class Mask, class Args = int>class PSkel::StencilBase< Array, Mask, Args >

Class that implements the basic functionalities supported by the stencil skeletons.

5.19.2 Member Function Documentation

5.19.2.1 template<class Array , class Mask , class Args > void PSkel::StencilBase< Array, Mask, Args >::runAutoGPU (size_t *GPUBlockSize* = 0)

Executes in GPU a single iteration of the stencil computation. If the data is larger than the memory available in the GPU, this function automatically selects a tiling execution of the stencil computation.

Parameters

in	*GPUBlock-Size*	the block size used for the GPU processing the stencil kernel. if GPUBlockSize is 0, the block size is automatically chosen.

5.19.2.2 template<class Array , class Mask , class Args > void PSkel::StencilBase< Array, Mask, Args >::runCPU (size_t *numThreads* = 0)

Executes in CPU, using multithreads, a single iteration of the stencil computation.

Parameters

in	*numThreads*	the number of threads used for processing the stencil kernel. if numThreads is 0, the number of threads is automatically chosen.

5.19.2.3 template<class Array , class Mask , class Args > void PSkel::StencilBase< Array, Mask, Args >::runGPU (size_t *GPUBlockSize* = 0)

Executes in GPU a single iteration of the stencil computation. This function does not handle data larger than the memory available in the GPU (see runAutoGPU.)

Parameters

in	*GPUBlock-Size*	the block size used for the GPU processing the stencil kernel. if GPUBlockSize is 0, the block size is automatically chosen.

5.19.2.4 template<class Array , class Mask , class Args > void PSkel::StencilBase< Array, Mask, Args >::runIterativeAutoGPU (size_t *iterations,* size_t *GPUBlockSize* = 0)

Executes in GPU multiple iterations of the stencil computation. At each given iteration, except the first, the previous output is used as input. If the data is larger than the memory available in the GPU, this function automatically selects a tiling execution of the stencil computation, including the number of iterations to be consecutivelly executed on GPU.

Parameters

in	*iterations*	the number of iterations to be computed.
in	*GPUBlock-Size*	the block size used for the GPU processing the stencil kernel. if GPUBlockSize is 0, the block size is automatically chosen.

5.19.2.5 template<class Array , class Mask , class Args > void PSkel::StencilBase< Array, Mask, Args >::runIterativeCPU (size_t *iterations,* size_t *numThreads* = 0)

Executes in CPU, using multithreads, multiple iterations of the stencil computation. At each given iteration, except the first, the previous output is used as input.

Parameters

in	*iterations*	the number of iterations to be computed.
in	*numThreads*	the number of threads used for processing the stencil kernel. if numThreads is 0, the number of threads is automatically chosen.

5.19.2.6 template<class Array , class Mask , class Args > void PSkel::StencilBase< Array, Mask, Args >::runIterativeGPU (size_t *iterations,* size_t *GPUBlockSize =* 0)

Executes in GPU multiple iterations of the stencil computation. At each given iteration, except the first, the previous output is used as input. This function does not handle data larger than the memory available in the GPU (see runIterativeAutoGPU.)

Parameters

in	*iterations*	the number of iterations to be computed.
in	*GPUBlock-Size*	the block size used for the GPU processing the stencil kernel. if GPUBlockSize is 0, the block size is automatically chosen.

5.19.2.7 template<class Array , class Mask , class Args > void PSkel::StencilBase< Array, Mask, Args >::runIterativeSequential (size_t *iterations*)

Executes sequentially in CPU multiple iterations of the stencil computation. At each given iteration, except the first, the previous output is used as input.

Parameters

in	*iterations*	the number of iterations to be computed.

5.19.2.8 template<class Array , class Mask , class Args > void PSkel::StencilBase< Array, Mask, Args >::runIterativeTilingGPU (size_t *iterations,* size_t *tilingWidth,* size_t *tilingHeight,* size_t *tilingDepth,* size_t *innerIterations =* 1, size_t *GPUBlockSize =* 0)

Executes in GPU multiple iterations of the stencil computation, tiling the input data. At each given iteration, except the first, the previous output is used as input. This function is useful for processing data larger than the memory available in the GPU (see runIterativeAutoGPU.)

Parameters

in	*iterations*	the number of iterations to be computed.
in	*tilingWidth*	the width size for each (logical) tile of the input data.
in	*tilingHeight*	the height size for each (logical) tile of the input data.

in	*tilingDepth*	the depth size for each (logical) tile of the input data.
in	*innerIterations*	the number of iterations to be consecutively executed on GPU; the number of iterations executed consecutively on increases the amount of memory required.
in	*GPUBlockSize*	the block size used for the GPU processing the stencil kernel. if GPUBlockSize is 0, the block size is automatically chosen.

5.19.2.9 template<class Array , class Mask , class Args > void PSkel::StencilBase< Array, Mask, Args >::runSequential ( )

Executes sequentially in CPU a single iteration of the stencil computation.

5.19.2.10 template<class Array , class Mask , class Args > void PSkel::StencilBase< Array, Mask, Args >::runTilingGPU (size_t *tilingWidth*, size_t *tilingHeight*, size_t *tilingDepth*, size_t *GPUBlockSize* = 0)

Executes in GPU a single iteration of the stencil computation, tiling the input data. This function is useful for processing data larger than the memory available in the GPU (see runAutoGPU.)

Parameters

in	*tilingWidth*	the width size for each (logical) tile of the input data.
in	*tilingHeight*	the height size for each (logical) tile of the input data.
in	*tilingDepth*	the depth size for each (logical) tile of the input data.
in	*GPUBlockSize*	the block size used for the GPU processing the stencil kernel. if GPUBlockSize is 0, the block size is automatically chosen.

The documentation for this class was generated from the following files:

- src/PSkelStencil.h
- src/PSkelStencil.hpp

5.20 PSkel::StencilTiling< Array, Mask > Class Template Reference

`#include <PSkelStencilTiling.h>`

Public Member Functions

- **StencilTiling** (Array in, Array out, Mask mask)

- void tile (size_t iterations, size_t widthOffset, size_t heightOffset, size_t depthOffset, size_t width, size_t height, size_t depth)

Public Attributes

- Mask **mask**
- Array **input**
- Array **output**
- size_t **iterations**
- size_t **widthOffset**
- size_t **heightOffset**
- size_t **depthOffset**
- size_t **width**
- size_t **height**
- size_t **depth**
- size_t **coreWidthOffset**
- size_t **coreHeightOffset**
- size_t **coreDepthOffset**
- size_t **coreWidth**
- size_t **coreHeight**
- size_t **coreDepth**

5.20.1 Detailed Description

template<class Array, class Mask>class PSkel::StencilTiling< Array, Mask >

Class that performs the necessary calculations -- regarding the halo region, tile size, etc. -- for tiling stencil computations.

5.20.2 Member Function Documentation

5.20.2.1 template<class Array, class Mask> void PSkel::StencilTiling< Array, Mask >::tile (size_t *iterations*, size_t *widthOffset*, size_t *heightOffset*, size_t *depthOffset*, size_t *width*, size_t *height*, size_t *depth*) [inline]

Updates the stencil tiling information for the specified number of iterations and tile size.

Parameters

in	*iterations*	number of iterations consecutively executed on the device (GPU).
in	*widthOffset*	width offset for the logical tile region, considering the input and output arrays.
in	*heightOffset*	height offset for the logical tile region, considering the input and output arrays.

in	*depthOffset*	depth offset for the logical tile region, considering the input and output arrays.
in	*width*	width of for the logical tile region.
in	*height*	height of the logical tile region.
in	*depth*	depth of the logical tile region.

The documentation for this class was generated from the following file:

- src/PSkelStencilTiling.h

5.21 PSkel::TilingGPUGeneticEvaluationFunction Struct Reference

Public Attributes

- size_t **iterations**
- size_t **height**
- size_t **width**
- size_t **depth**
- size_t **range**
- size_t **typeSize**
- size_t **memFree**
- size_t **popsize**
- size_t **ngen**
- size_t **dw**
- size_t **dt**
- size_t **dh**
- float **score**

The documentation for this struct was generated from the following file:

- src/PSkelStencil.hpp

Chapter 6

File Documentation

6.1 src/PSkel.h File Reference

```
#include <stdio.h>  #include <cstdio>  #include <stdlib.-
h> #include <time.h> #include <typeinfo> #include <iostream> ×
#include <cuda.h>  #include <cuda_runtime_api.h>  #include
<tbb/blocked_range.h>     #include <tbb/parallel_for.h> ×
#include <tbb/task_scheduler_init.h>   #include <omp.h> ×
#include "PSkelDefs.h"  #include "PSkelArray.h"  #include
"PSkelArgs.h"   #include "PSkelMask.h"   #include "PSkel-
Stencil.h"
```

6.1.1 Detailed Description

This file contains all the includes required for using the PSkel framework.

6.2 src/PSkelArray.h File Reference

```
#include <cuda.h> #include "PSkelDefs.h" #include "PSkel-
Array.hpp"
```

Classes

- class PSkel::ArrayBase< T >
- class PSkel::Array3D< T >
- class PSkel::Array2D< T >
- class PSkel::Array< T >

6.2.1 Detailed Description

This file contains the definition for the arrays data structures in PSkel.

6.3 src/PSkelDefs.h File Reference

`#include <stdio.h>  #include <cuda.h>  #include <cuda_runtime_api.h>`

Defines

- #define **__parallel__** __device__ __host__ __attribute__((always_inline)) __forceinline__
- #define **gpuErrchk**(ans) { gpuAssert(((cudaError_t)ans), __FILE__, __LINE__,true); }

Functions

- void **gpuAssert** (cudaError_t code, const char *file, int line, bool abort=false)

6.3.1 Detailed Description

This file contains basic macro definitions used throughout PSkel.

6.4 src/PSkelMask.h File Reference

`#include "PSkelArray.h" #include "PSkelMask.hpp"`

Classes

- class PSkel::MaskBase< T >
- class PSkel::Mask3D< T >
- class PSkel::Mask2D< T >
- class PSkel::Mask< T >

6.4.1 Detailed Description

This file contains the definition for the mask data structures, which is used by the stencil skeleton.

6.5 src/PSkelStencil.h File Reference

```
#include "PSkelDefs.h" #include "PSkelArray.h" #include "-
PSkelMask.h" #include "PSkelStencilTiling.h" #include "P-
SkelStencil.hpp"
```

Classes

- class PSkel::StencilBase< Array, Mask, Args >
- class PSkel::Stencil3D< Array, Mask, Args >
- class PSkel::Stencil2D< Array, Mask, Args >
- class PSkel::Stencil< Array, Mask, Args >

Functions

- template<typename T1 , typename T2 , class Args >
  __parallel__ void **PSkel::stencilKernel** (Array< T1 > input, Array< T1 > output, Mask< T2 > mask, Args args, size_t i)
- template<typename T1 , typename T2 , class Args >
  __parallel__ void **PSkel::stencilKernel** (Array2D< T1 > input, Array2D< T1 > output, Mask2D< T2 > mask, Args args, size_t h, size_t w)
- template<typename T1 , typename T2 , class Args >
  __parallel__ void **PSkel::stencilKernel** (Array3D< T1 > input, Array3D< T1 > output, Mask3D< T2 > mask, Args args, size_t h, size_t w, size_t d)

6.5.1 Detailed Description

This file contains the definition for the stencil skeleton.