SHAN

Generated by Doxygen 1.8.11

Contents

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

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

Chapter 3

Class Documentation

3.1 shan_element_t Struct Reference

`#include <SHAN_comm.h>`

Public Attributes

- long maxSendSz
  max send size per type (byte)
- long maxRecvSz
  max recv size per type (byte)
- int max_nelem_send
  max recv size per type (byte)
- int max_nelem_recv
  max recv size per type (byte)
- long SendOffset [2]
  local offset for send per type (byte)
- long RecvOffset [2]
  local offset for recv per type (byte)
- long elemOffset
  element offset in shared mem
- int * local_send_count
  send stage counter array, per type
- int * local_recv_count
  recv stage counter array, per type
- int * local_ack_count
  acknowledge stage counter array, per type

3.1.1 Detailed Description

comm struct, holds all communication information per type.

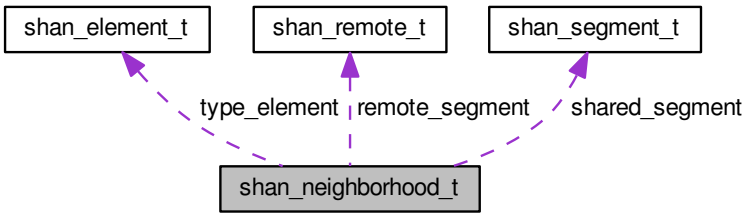
The documentation for this struct was generated from the following file:

- include/SHAN_comm.h

3.2 shan_neighborhood_t Struct Reference

`#include <SHAN_comm.h>`

Collaboration diagram for shan_neighborhood_t:



Public Attributes

- int neighbor_hood_id
  neighborhood id
- MPI_Comm MPI_COMM_SHM
  shared MPI communicator
- MPI_Comm MPI_COMM_ALL
  global MPI communicator
- int num_neighbors
  num comm partners (neighbors)
- int num_local
  node local number of comm partners
- int * neighbors
  list of neighbors, per rank
- int * RemoteCommIndex
  the remote index corresponding to own rank
- int * RemoteNumNeighbors
  remote number of neighbors for RemoteCommIndex
- int num_type
  num types
- long * typeOffset
  type offsets for all node local ranks
- shan_segment_t shared_segment
  shared window for local communication
- shan_element_t * type_element
  local comm data for remote communication.
- long remoteSz
  remote comm size, all types, send + recv (byte)
- shan_remote_t remote_segment
  private segment for remote communication
- int nProcLocal

num local ranks in shared mem

- int iProcLocal

  local rank id
- int nProcGlobal

  num global ranks
- int iProcGlobal

  global rank id
- int master

  master of shared segment (local rank 0)
- int * remote_master

  global list of masters
- int * local_stage_count

  generic stage counter for wait4All(Send/Recv)

3.2.1 Detailed Description

neighborhood comm struct, holds all communication information for the neighborhood.

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

- include/SHAN_comm.h

3.3 shan_notification_t Struct Reference

```
#include <SHAN_segment.h>
```

Public Member Functions

- volatile int val __attribute__ ((aligned(64)))

  notification value

3.3.1 Detailed Description

64 byte aligned notifications struct for shared mem notifications.

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

- include/SHAN_segment.h

3.4 shan_remote_t Struct Reference

```
#include <SHAN_comm.h>
```

Public Attributes

- int shan_id
  shared segment id
- long dataSz
  segment size array
- void * shan_ptr
  local segment pointer

3.4.1 Detailed Description

Segment struct, rank_local, holds all segment information.

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

- include/SHAN_comm.h

3.5 shan_segment_t Struct Reference

```
#include <SHAN_segment.h>
```

Public Attributes

- void **restrict ptr_array
  shan ptr array
- int shan_id
  shared segment id
- int shan_type
  shared segment type
- long dataSz
  segment size
- long * localDataSz
  segment size array
- MPI_Comm MPI_COMM_SHM
  MPI shared mem communicator.
- int * fd
  shmem file descriptor array
- char shan_domain_name [80]
  unique shmem name

3.5.1 Detailed Description

Segment struct, shared, holds all segment information.

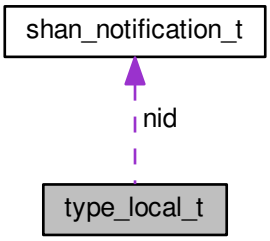
The documentation for this struct was generated from the following file:

- include/SHAN_segment.h

3.6 type_local_t Struct Reference

`#include <SHAN_comm.h>`

Collaboration diagram for type_local_t:



Public Attributes

- shan_notification_t * nid
  synchronization for types
- int * nelem_send
  current num send elements per neighbor
- int * nelem_recv
  current num recv elements per neighbor
- int * send_sz
  current send size (in char) per neighbor
- int * recv_sz
  current recv size (in char) per neighbor
- long * send_offset
  list of send offsets per neighbor
- long * recv_offset
  list of recv offsets per neighbor

3.6.1 Detailed Description

Type struct, visible in shared memory

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

- include/SHAN_comm.h

Chapter 4

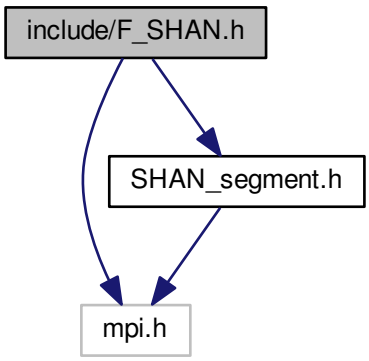
File Documentation

4.1 include/F_SHAN.h File Reference

Wrapper functions for the SHAN library, mostly targeted at fortran applications.

```
#include <mpi.h>
#include "SHAN_segment.h"
```

Include dependency graph for F_SHAN.h:



Functions

- void f_shan_alloc_shared (const int segment_id, const long dataSz, void **restrict shm_ptr)
- void f_shan_free_shared (const int segment_id)
- void f_shan_free_comm (const int neighbor_hood_id)
- void f_shan_init_comm (const int neighbor_hood_id, void *neighbors, int num_neighbors, void *maxSendSz, void *maxRecvSz, void *max_nelem_send, void *max_nelem_recv, int num_type)
- void f_shan_type_offset (const int neighbor_hood_id, const int type_id, void **nelem_send, void **nelem↩ _recv, void **send_sz, void **recv_sz, void **send_idx, void **recv_idx)
- void f_shan_comm_wait4All (const int neighbor_hood_id, const int segment_id, const int type_id)
- void f_shan_comm_wait4AllSend (const int neighbor_hood_id, const int type_id)
- void f_shan_comm_wait4AllRecv (const int neighbor_hood_id, const int segment_id, const int type_id)
- void f_shan_comm_notify_or_write (const int neighbor_hood_id, const int segment_id, const int type_id, int idx)

4.1.1 Detailed Description

Wrapper functions for the SHAN library, mostly targeted at fortran applications.

4.1.2 Function Documentation

4.1.2.1 void f_shan_alloc_shared (const int *segment_id,* const long *dataSz,* void ∗∗restrict *shm_ptr*)

wrapper function for shan_alloc_shared

Note: Memory will be page-aligned.

Parameters

segment_id	- segment handle (data)
dataSz	- segment size in byte
shm_ptr	- shared mem pointer for allocated memory

Returns

SHAN_SUCCESS in case of success, SHAN_ERROR in case of error.

4.1.2.2 void f_shan_comm_notify_or_write (const int *neighbor_hood_id,* const int *segment_id,* const int *type_id,* int *idx*)

wrapper function for shan_comm_notify_or_write

Parameters

neighbor_hood_id	- general neighborhood handle
segment_id	- data segment handle
type_id	- used type id
idx	- comm index for target rank in neighborhood

Returns

SHAN_SUCCESS in case of success, SHAN_ERROR in case of error.

4.1.2.3 void f_shan_comm_wait4All (const int *neighbor_hood_id,* const int *segment_id,* const int *type_id*)

wrapper function for shan_comm_wait4All

Parameters

neighbor_hood↩ _id	- general neighborhood handle
segment_id	- (data) segment handle
type_id	- used type id

Returns

SHAN_SUCCESS in case of success, SHAN_ERROR in case of error.

4.1.2.4 void f_shan_comm_wait4AllRecv (const int *neighbor_hood_id,* const int *segment_id,* const int *type_id*)

wrapper function for shan_comm_wait4AllRecv

Parameters

neighbor_hood↩ _id	- general neighborhood handle
segment_id	- (data) segment handle
type_id	- used type id

Returns

SHAN_SUCCESS in case of success, SHAN_ERROR in case of error.

4.1.2.5 void f_shan_comm_wait4AllSend (const int *neighbor_hood_id,* const int *type_id*)

wrapper function for shan_comm_wait4AllSend

Parameters

neighbor_hood↩ _id	- general neighborhood handle
type_id	- used type id

Returns

SHAN_SUCCESS in case of success, SHAN_ERROR in case of error.

4.1.2.6 void f_shan_free_comm (const int *neighbor_hood_id*)

wrapper function for shan_free_comm

Parameters

neighbor_hood↩ _id	- general neighborhood handle

Returns

SHAN_SUCCESS in case of success, SHAN_ERROR in case of error.

4.1.2.7 void f_shan_free_shared (const int *segment_id*)

wrapper function for shan_free_shared

Parameters

segment↩ _id	- segment handle (data)

Returns

SHAN_SUCCESS in case of success, SHAN_ERROR in case of error.

4.1.2.8 void f_shan_init_comm (const int *neighbor_hood_id,* void * *neighbors,* int *num_neighbors,* void * *maxSendSz,* void * *maxRecvSz,* void * *max_nelem_send,* void * *max_nelem_recv,* int *num_type*)

wrapper function for shan_init_comm

Parameters

neighbor_hood↩ _id	- general neighborhood handle
neighbors	- comm partners (neighbors)
num_neighbors	- num comm partners (neighbors)
maxSendSz	- max send size for every comm type (byte)
maxRecvSz	- max recv size for every comm type (byte)
max_nelem_send	- max number of send elements for every comm type
max_nelem_recv	- max number of recv elements for every comm type
num_type	- number of types

Returns

SHAN_SUCCESS in case of success, SHAN_ERROR in case of error.

4.1.2.9 void f_shan_type_offset (const int *neighbor_hood_id,* const int *type_id,* void ** *nelem_send,* void ** *nelem_recv,* void ** *send_sz,* void ** *recv_sz,* void ** *send_idx,* void ** *recv_idx*)

wrapper function for shan_type_offset

Parameters

neighbor_hood↩ _id	- general neighborhood handle
type_id	- used type segment
nelem_send	- ptr for current number of send elements. (in shared mem, visible node locally)
nelem_recv	- ptr for current number of recv elements. (in shared mem, visible node locally)
send_sz	- ptr for current send_size. (in shared mem, visible node locally)
recv_sz	- ptr for current recv size. (in shared mem, visible node locally)
send_idx	- ptr for current send offset list. (in shared mem, visible node locally)
recv_idx	- ptr for current recv offset list. (in shared mem, visible node locally)

Returns

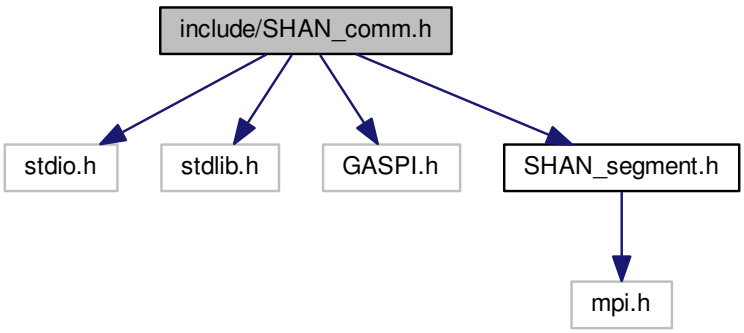
SHAN_SUCCESS in case of success, SHAN_ERROR in case of error.

4.2 include/SHAN_comm.h File Reference

SHAN_comm header for persistant communication in shared memory.

```
#include <stdio.h>
#include <stdlib.h>
#include "GASPI.h"
#include "SHAN_segment.h"
```

Include dependency graph for SHAN_comm.h:



Classes

- struct type_local_t
- struct shan_remote_t
- struct shan_element_t
- struct shan_neighborhood_t

Macros

- #define MAX_SHARED_NOTIFICATION 2

  'have written' and 'have read' synchronization

Functions

- int shan_comm_local_rank (shan_neighborhood_t *const neighborhood_id, const int rank)
- int shan_increment_local (shan_neighborhood_t *const neighborhood_id, int const type_id, int const idx)
- int shan_comm_init_comm (shan_neighborhood_t *const neighborhood_id, int neighbor_hood_id, int *neighbors, int num_neighbors, long *maxSendSz, long *maxRecvSz, int *max_nelem_send, int *max_nelem_recv, int num_type, MPI_Comm MPI_COMM_SHM, MPI_Comm MPI_COMM_ALL)
- int shan_comm_free_comm (shan_neighborhood_t *const neighborhood_id)
- int shan_comm_notify_or_write (shan_neighborhood_t *const neighborhood_id, shan_segment_t *data_segment, int type_id, int idx)
- int shan_comm_wait4All (shan_neighborhood_t *const neighborhood_id, shan_segment_t *data_segment, int type_id)
- int shan_comm_wait4AllRecv (shan_neighborhood_t *const neighborhood_id, shan_segment_t *data_segment, int type_id)
- int shan_comm_wait4Recv (shan_neighborhood_t *const neighborhood_id, shan_segment_t *data_segment, int type_id, int idx)
- int shan_comm_test4Recv (shan_neighborhood_t *const neighborhood_id, shan_segment_t *data_segment, int type_id, int idx)
- int shan_comm_wait4AllSend (shan_neighborhood_t *const neighborhood_id, int type_id)
- int shan_comm_wait4Send (shan_neighborhood_t *const neighborhood_id, int type_id, int idx)
- int shan_comm_test4Send (shan_neighborhood_t *const neighborhood_id, int type_id, int idx)
- void shan_comm_shmemBarrier (shan_neighborhood_t *const neighborhood_id)

4.2.1 Detailed Description

SHAN_comm header for persistant communication in shared memory.

4.2.2 Function Documentation

4.2.2.1 int shan_comm_free_comm (shan_neighborhood_t *const *neighborhood_id*)

Free communication ressources

Parameters

neighborhood_id	- general neighborhood handle

Returns

SHAN_COMM_SUCCESS in case of success, SHAN_COMM_ERROR in case of error.

4.2.2.2 int shan_comm_init_comm (shan_neighborhood_t *const *neighborhood_id,* int *neighbor_hood_id,* int * *neighbors,* int *num_neighbors,* long * *maxSendSz,* long * *maxRecvSz,* int * *max_nelem_send,* int * *max_nelem_recv,* int *num_type,* MPI_Comm *MPI_COMM_SHM,* MPI_Comm *MPI_COMM_ALL*)

Initialize persistant communication for shared mem and GASPI. requires bidirectional communication for synchronization in one-sided communication.

A zero length messages will work, no message at all will fail.

- allocates shared and private mem for communication (double buffered).
- figures out local and remote comm partners.
- negotiates remote number f neighbors and comm index

Parameters

neighborhood_id	- general neighborhood handle
neighbor_hood_id	- neighborhood id
neighbors	- comm partners (neighbors)
num_neighbors	- num comm partners (neighbors)
maxSendSz	- max send size for every type.
maxRecvSz	- max recv size for every type
max_nelem_send	- max number of send elements per type
max_nelem_recv	- max number of recv elements per type
num_type	- number of types
MPI_COMM_SHM	- MPI shared mem communicator
MPI_COMM_ALL	- embedding of shared communicator (typically MPI_COMM_WORLD)

Returns

SHAN_COMM_SUCCESS in case of success, SHAN_COMM_ERROR in case of error.

4.2.2.3 int shan_comm_local_rank (shan_neighborhood_t *const *neighborhood_id,* const int *rank*)

Gets node local rank id.

Parameters

neighborhood_id	- handle for neighborhood
rank	- global rank

Returns

SHAN_COMM_SUCCESS in case of success, SHAN_COMM_ERROR in case of error.

4.2.2.4 int shan_comm_notify_or_write (shan_neighborhood_t *const *neighborhood_id,* shan_segment_t * *data_segment,* int *type_id,* int *idx*)

Writes data or flags data as readable.

- aggregates send data into linear buffer or
- flags data as readable
  - number of elements
  - element sizes and
  - element offsets

Parameters

neighborhood_id	- general neighborhood handle
data_segment	- data segment handle
type_id	- type index
idx	- comm index for target rank in neighborhood

Returns

SHAN_COMM_SUCCESS in case of success, SHAN_COMM_ERROR in case of error.

4.2.2.5 void shan_comm_shmemBarrier (shan_neighborhood_t *const *neighborhood_id*)

Shared mem barrier

Parameters

neighborhood_id	- general neighborhood handle

4.2.2.6 int shan_comm_test4Recv (shan_neighborhood_t *const *neighborhood_id,* shan_segment_t * *data_segment,* int *type_id,* int *idx*)

Tests for specific receive requests.

Parameters

neighborhood_id	- general neighborhood handle
data_segment	- data segment handle
type_id	- type index
idx	- comm index for target rank in neighborhood

Returns

SHAN_COMM_SUCCESS in case of success, SHAN_COMM_ERROR in case of error.

4.2.2.7 int shan_comm_test4Send (shan_neighborhood_t *const *neighborhood_id,* int *type_id,* int *idx*)

Tests for specific send requests

Parameters

neighborhood_id	- general neighborhood handle
type_id	- type index
idx	- comm index for target rank in neighborhood

Returns

SHAN_COMM_SUCCESS in case of success, SHAN_COMM_ERROR in case of error.

4.2.2.8 int shan_comm_wait4All (shan_neighborhood_t *const *neighborhood_id,* shan_segment_t * *data_segment,* int *type_id*)

Waits for entire neighborhood

- waits for either shared memory notifications or remote GASPI notifications
- directly converts send type into recv type in shared memory
- unpacks pipelined remote communictation into the current receive type.
- waits for all receive requests.
- waits for all send requests

Parameters

neighborhood_id	- general neighborhood handle
data_segment	- data segment handle
type_id	- type index

Returns

SHAN_COMM_SUCCESS in case of success, SHAN_COMM_ERROR in case of error.

4.2.2.9 int shan_comm_wait4AllRecv (shan_neighborhood_t *const *neighborhood_id,* shan_segment_t * *data_segment,* int *type_id*)

Waits for entire neighborhood

- waits for either shared memory notifications or remote GASPI notifications
- directly converts send type into recv type in shared memory
- unpacks pipelined remote communictation into the current receive type.
- waits for all receive requests.

Parameters

neighborhood_id	- general neighborhood handle
data_segment	- data segment handle
type_id	- type index

Returns

SHAN_COMM_SUCCESS in case of success, SHAN_COMM_ERROR in case of error.

4.2.2.10 int shan_comm_wait4AllSend (shan_neighborhood_t *const *neighborhood_id,* int *type_id*)

Waits for entire neighborhood

- waits for all send requests

Parameters

neighborhood_id	- general neighborhood handle
type_id	- type index

Returns

SHAN_COMM_SUCCESS in case of success, SHAN_COMM_ERROR in case of error.

4.2.2.11 int shan_comm_wait4Recv (shan_neighborhood_t *const *neighborhood_id,* shan_segment_t * *data_segment,* int *type_id,* int *idx*)

waits for specific receive requests.

Parameters

neighborhood_id	- general neighborhood handle
data_segment	- data segment handle
type_id	- type index
idx	- comm index for target rank in neighborhood

Returns

SHAN_COMM_SUCCESS in case of success, SHAN_COMM_ERROR in case of error.

4.2.2.12 int shan_comm_wait4Send (shan_neighborhood_t *const *neighborhood_id,* int *type_id,* int *idx*)

Waits for specific send requests

Parameters

neighborhood_id	- general neighborhood handle
type_id	- type index
idx	- comm index for target rank in neighborhood

Returns

SHAN_COMM_SUCCESS in case of success, SHAN_COMM_ERROR in case of error.

4.2.2.13 int shan_increment_local (shan_neighborhood_t *const *neighborhood_id,* int const *type_id,* int const *idx*)

Increments counter in shared mem

Parameters

neighborhood_id	- general neighborhood handle
type_id	- type index
idx	- comm index for target rank in neighborhood

Returns

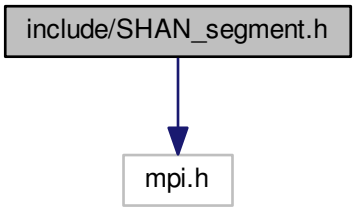
SHAN_COMM_SUCCESS in case of success, SHAN_COMM_ERROR in case of error.

4.3 include/SHAN_segment.h File Reference

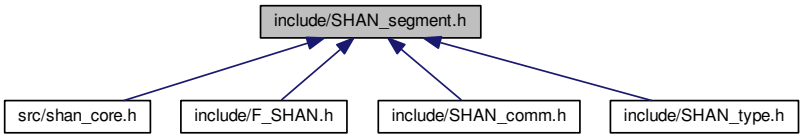
SHAN_segment header for notifications in shared memory.

```
#include <mpi.h>
```

Include dependency graph for SHAN_segment.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct shan_notification_t
- struct shan_segment_t

Enumerations

- enum **shan_type** { **SHAN_DATA** = 0, **SHAN_TYPE** = 1 }
- enum **shan_return_val** { **SHAN_ERROR** = -2, **SHAN_FAIL** = -1, **SHAN_SUCCESS** = 0 }

Functions

- int shan_alloc_shared (shan_segment_t ∗const segment, const int shan_id, const int shan_type, const long dataSz, const MPI_Comm MPI_COMM_SHM)
- int shan_free_shared (shan_segment_t ∗const segment)
- int shan_get_shared_ptr (shan_segment_t ∗const segment, const int rank, void ∗∗shm_ptr)
- int shan_notify_reset_shared (shan_notification_t ∗const ptr, const int idx, int ∗const val)
- int shan_notify_increment_shared (shan_notification_t ∗const ptr, const int idx, const int increment)
- int shan_notify_test_shared (shan_notification_t ∗const ptr, const int idx, int ∗const val)
- int shan_notify_init_shared (shan_notification_t ∗const ptr, const int idx)

4.3.1 Detailed Description

SHAN_segment header for notifications in shared memory.

The SHAN (SHA_red N_otifications) interface is a user-level API which aims at migrating flat MPI (legacy) code towards an asynchronous dataflow model. SHAN uses the GASPI API and extends ideas from MPI shared windows.

GASPI is a PGAS communication library which is based on the concept of one-sided, notified communication. The synchronization context here is bundled together with a one-sided message such that a communication target becomes able to test for completion of the received one-sided communication.

Traditionally the GASPI programming model has been aimed at multithreaded or task-based applications. In G↩ ASPI the synchronization context is bundled together with a one-sided message such that a communication target becomes able to test for completion of the received one-sided communication.

In order to support a migration of legacy applications (with a flat MPI communication model) towards GASPI, we have extended the concept of shared MPI windows towards a notified communication model in which the processes sharing a common window become able to see all one-sided and notified communication targeted at this window. Similarly we have extended the concept of MPI shared windows with shared notifications, which are globally visible in shared memory.

Besides the possibility to entirely avoid node-internal communication and to make use of a much improved overlap of communication and computation the model of notified communication in GASPI shared windows will allow legacy SPMD applications a transition towards an asynchronous dataflow model.

4.3.2 Function Documentation

4.3.2.1 int shan_alloc_shared (shan_segment_t *const *segment*, const int *shan_id*, const int *shan_type*, const long *dataSz*, const MPI_Comm *MPI_COMM_SHM*)

Local allocation of shared memory of size dataSz

Note: Memory will be page-aligned.

Parameters

segment	- segment handle
shan_id	- (unique) segment id
shan_type	- type of allocated memory
dataSz	- required memory size per rank in byte
MPI_COMM_SHM	- shared mem communicator

Returns

SHAN_SUCCESS in case of success, SHAN_ERROR in case of error.

4.3.2.2 int shan_free_shared (shan_segment_t *const *segment*)

Free shared memory.

Parameters

segment	- segment handle

Returns

SHAN_SUCCESS in case of success, SHAN_ERROR in case of error.

4.3.2.3 int shan_get_shared_ptr (shan_segment_t *const *segment*, const int *rank*, void ** *shm_ptr*)

Gets shared mem pointer for node local ranks

Parameters

segment	- segment handle
rank	- node local rank id
shm_ptr	- required memory size per rank in byte

Returns

SHAN_SUCCESS in case of success, SHAN_ERROR in case of error.

4.3.2.4 int shan_notify_increment_shared (shan_notification_t *const *ptr*, const int *idx*, const int *increment*)

Increments shared mem notfication. Sets write fence such that local result is valid, once the incremented value is visible for other local ranks.

Parameters

ptr	- pointer to shared notification array
idx	- shared mem notification id
increment	- increment value

Returns

SHAN_SUCCESS in case of success, SHAN_ERROR in case of error.

4.3.2.5 int shan_notify_init_shared (shan_notification_t *const *ptr*, const int *idx*)

Tests for shared mem notfication.

Parameters

ptr	- pointer to shared notification array
idx	- shared mem notification id

Returns

SHAN_SUCCESS in case of success, SHAN_ERROR in case of error.

4.3.2.6 int shan_notify_reset_shared (shan_notification_t ∗const *ptr,* const int *idx,* int ∗const *val*)

Resets shared mem notification.

Parameters

ptr	- pointer to shared notification array
idx	- shared mem notification id
val	- old value of the notification

Returns

SHAN_SUCCESS in case of success, SHAN_ERROR in case of error.

4.3.2.7 int shan_notify_test_shared (shan_notification_t ∗const *ptr,* const int *idx,* int ∗const *val*)

Tests for shared mem notfication.

Parameters

ptr	- pointer to shared notification array
idx	- shared mem notification id
val	- current notification value

Returns

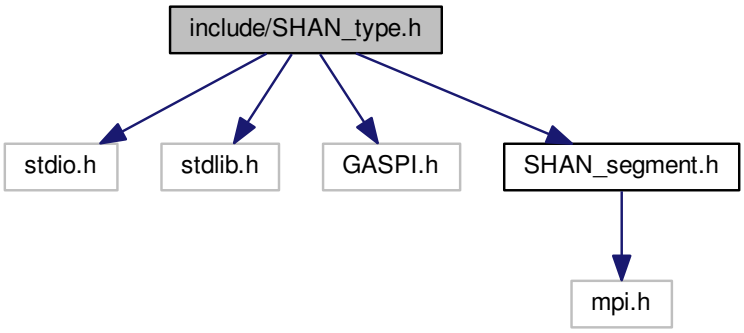
SHAN_SUCCESS in case of success, SHAN_ERROR in case of error.

4.4 include/SHAN_type.h File Reference

SHAN_type header. Type conversion in shared memory.

```
#include <stdio.h>
#include <stdlib.h>
#include "GASPI.h"
#include "SHAN_segment.h"
```

Include dependency graph for SHAN_type.h:



Functions

- void shan_comm_get_local (shan_neighborhood_t *neighborhood_id, shan_segment_t *data_segment, const int type_id, const int idx)
- void shan_comm_get_remote (shan_neighborhood_t *neighborhood_id, shan_segment_t *const data_segment, int const type_id, int const idx)
- int shan_get_shared_type (type_local_t *type_info, shan_neighborhood_t *neighborhood_id, int local_rank, int num_neighbors, int type_id)
- int shan_comm_type_offset (shan_neighborhood_t *neighborhood_id, int type_id, int **nelem_send, int **nelem_recv, int **send_sz, int **recv_sz, long **send_offset, long **recv_offset)
- int shan_comm_get_type (type_local_t *type_info, void *shm_ptr, int num_neighbors, shan_element_t *type_element)

4.4.1 Detailed Description

SHAN_type header. Type conversion in shared memory.

4.4.2 Function Documentation

4.4.2.1 void shan_comm_get_local (shan_neighborhood_t * *neighborhood_id*, shan_segment_t * *data_segment*, const int *type_id*, const int *idx*)

Converts shared mem send type in shared mem recv type. Finalizes receive in shared mem.

Parameters

neighborhood_id	- general neighborhood handle
data_segment	- used data segment
type_id	- used type id
idx	- rank index in neighborhood

Returns

SHAN_COMM_SUCCESS in case of success, SHAN_COMM_ERROR in case of error.

4.4.2.2 void shan_comm_get_remote (shan_neighborhood_t * *neighborhood_id,* shan_segment_t *const *data_segment,* int const *type_id,* int const *idx*)

Finalizes receive for remote comm

Parameters

neighborhood_id	- general neighborhood handle
data_segment	- used data segment
type_id	- used type id
idx	- rank index in neighborhood

Returns

SHAN_COMM_SUCCESS in case of success, SHAN_COMM_ERROR in case of error.

4.4.2.3 int shan_comm_get_type (type_local_t * *type_info,* void * *shm_ptr,* int *num_neighbors,* shan_element_t * *type_element*)

Getter function for type data

Parameters

type_info	- type data struct (SHAN_comm.h)
shm_ptr	- pointer to shared memory
num_neighbors	- rank local number of neighbors in neighborhood
type_element	- type element

Returns

SHAN_COMM_SUCCESS in case of success, SHAN_COMM_ERROR in case of error.

4.4.2.4 int shan_comm_type_offset (shan_neighborhood_t * *neighborhood_id,* int *type_id,* int ** *nelem_send,* int ** *nelem_recv,* int ** *send_sz,* int ** *recv_sz,* long ** *send_offset,* long ** *recv_offset*)

Gets type data structure for node local ranks Convenience call, as explicitly requesting remote type information should be rare.

Parameters

neighborhood_id	- general neighborhood handle

Parameters

type_id	- used type id
nelem_send	- pointer to number of send elements in shared mem
nelem_recv	- pointer to number of recv elements in shared mem
send_sz	- pointer to send size in shared mem
recv_sz	- pointer to recv size in shared mem
send_offset	- pointer to offset of send elements in shared mem
recv_offset	- pointer to offset of recv elements in shared mem

Returns

SHAN_COMM_SUCCESS in case of success, SHAN_COMM_ERROR in case of error.

4.4.2.5 int shan_get_shared_type (type_local_t * *type_info*, shan_neighborhood_t * *neighborhood_id*, int *local_rank*, int *num_neighbors*, int *type_id*)

Returns type data structure for node local ranks

Parameters

type_info	- type data struct (SHAN_comm.h)
neighborhood_id	- general neighborhood handle
local_rank	- node local rank
num_neighbors	- number of neighbors
type_id	- used type id

Returns

SHAN_COMM_SUCCESS in case of success, SHAN_COMM_ERROR in case of error.

Index