RDMA_CS

Generated by Doxygen 1.8.5

Fri May 19 2017 16:11:40

Contents

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

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

Chapter 3

Data Structure Documentation

3.1 client Struct Reference

Node for a linked lists containing information about open memory regions.

`#include <rdma_cs.h>`

Data Fields

- unsigned long cid
- uint32_t rkey
- uint64_t remote_addr
- size_t length
- struct client * next

3.1.1 Detailed Description

Node for a linked lists containing information about open memory regions.

3.1.2 Field Documentation

3.1.2.1 unsigned long cid

The numerical identification number of the client that owns the memory region

3.1.2.2 size_t length

The length of the memory region

3.1.2.3 struct client* next

A pointer to the next node in the list

3.1.2.4 uint64_t remote_addr

The address on the server of the memory region

3.1.2.5 uint32_t rkey

The rkey associated with the memory region

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

- rdma_cs.h

3.2 cnode Struct Reference

Linked list node containing information on all connected clients.

Data Fields

- struct rdma_cm_id * id
- pthread_t tid
- unsigned long cid
- uint32_t rkey
- uint64_t remote_addr
- size_t length
- enum client_status status
- struct cnode * next

3.2.1 Detailed Description

Linked list node containing information on all connected clients.

3.2.2 Field Documentation

3.2.2.1 unsigned long cid

The numerical id

3.2.2.2 struct rdma_cm_id* id

The communication manager id

3.2.2.3 size_t length

The length of the server-side memory region

3.2.2.4 struct cnode* next

A pointer to the next node in the list

3.2.2.5 uint64_t remote_addr

The address of the server-side memory region

3.2.2.6 uint32_t rkey

The rkey of the server-side memory region

3.2.2.7 enum client_status status

The status of the server-side memory region

3.2.2.8 pthread_t tid

The thread id

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

- server.c

3.3 pnode Struct Reference

Linked list node containing information on all running threads.

Data Fields

- pthread_t id
- unsigned short type
- struct pnode * next

3.3.1 Detailed Description

Linked list node containing information on all running threads.

3.3.2 Field Documentation

3.3.2.1 pthread_t id

The thread id

3.3.2.2 struct pnode* next

A pointer to the next node in the list

3.3.2.3 unsigned short type

The purpose of the thread

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

- server.c

Chapter 4

File Documentation

4.1 client.c File Reference

A RDMA client.

```
#include "rdma_cs.h"
```

Functions

- void connect_four (struct rdma_cm_id ∗cm_id, struct rdma_event_channel ∗ec, char ∗ip, short int port)

  Connect to a server at the given ip and port.
- void ∗ server_com (void ∗info)

  Listen for messages from the server.
- void add_client (struct client node)

  Add information about an open memory region to the list.
- void remove_client (unsigned long id)

  Remove information about an open memory region from the list.
- struct client ∗ get_client ()

  the get cm_id of the desired open memory region
- int **main** (int argc, char ∗∗argv)

Variables

- struct client ∗ clist_head =NULL

  The head of the list containing information on all open memory regions on the server.
- unsigned long clients = 0

  The current number of open memory regions.
- char ∗ menu1

  The first menu (operations on this client's remote memory region)
- char ∗ menu2

  The second menu (operations on other clients' remote memory regions)
- unsigned char in_menu

  Allows the communication thread to know if the main thread is in the menu (for re-printing the menu, if necessary)

4.1.1 Detailed Description

A RDMA client. This is the RDMA client that is to be paired with the server.c file.

Author

Austin Pohlmann

4.1.2 Function Documentation

4.1.2.1 void add_client (struct client *node*)

Add information about an open memory region to the list.

Returns

NULL

Parameters

node	the node to add to the list

4.1.2.2 void connect_four (struct rdma_cm_id * *cm_id,* struct rdma_event_channel * *ec,* char * *ip,* short int *port*)

Connect to a server at the given ip and port.

Returns

NULL

Parameters

cm_id	the cm_id associated with this client
ec	the event channel to use
ip	the ip to connect to
port	the port to connect to

4.1.2.3 struct client * get_client ( )

the get cm_id of the desired open memory region

Returns

the `struct client` node of the chosen memory region

4.1.2.4 void remove_client (unsigned long *id*)

Remove information about an open memory region from the list.

Returns

NULL

Parameters

id	the client's id number associated with the memory region to be removed

4.1.2.5 void ∗ server_com (void ∗ *info*)

Listen for messages from the server.

Returns

`NULL`

Parameters

info	a `struct listen_info` object with the needed information

4.1.3 Variable Documentation

4.1.3.1 char∗ menu1

Initial value:

```
= "----------------------\n"
            "| RDMA client         |\n"
            "----------------------\n"
            "1) Disconnect         |\n"
            "2) Write inline       |\n"
            "3) Write              |\n"
            "4) Read               |\n"
            "5) Open Server MR     |\n"
            "6) Close server MR    |\n"
```

The first menu (operations on this client's remote memory region)

4.1.3.2 char∗ menu2

Initial value:

```
= "----------------------\n"
            "| RDMA client(page 2)|\n"
            "----------------------\n"
            "1) Write inline       |\n"
            "2) Write              |\n"
            "3) Read               |\n"
            "4) Go back            |\n"
```

The second menu (operations on other clients' remote memory regions)

4.2 rdma_cs.c File Reference

File containing the definitions of the functions listed in rdma_cs.h.

```
#include "rdma_cs.h"
```

Functions

- void stop_it (char ∗reason, int error, FILE ∗file)

Print an error message and exit the application.

- struct rdma_cm_id * cm_event (struct rdma_event_channel *ec, enum rdma_cm_event_type expected, FILE *file)

  Process a communication manager event.
- void swap_info (struct rdma_cm_id *cm_id, struct ibv_mr *mr, uint32_t *rkey, uint64_t *remote_addr, size_t *size, FILE *file)

  Exchange the information needed to perform rdma read/write operations.
- uint32_t get_completion (struct rdma_cm_id *cm_id, enum completion_type type, uint8_t print, FILE *file)

  Wait for and pull a work completion.
- int obliterate (struct rdma_cm_id *mine, struct rdma_cm_id *client, struct ibv_mr *mr, struct rdma_event_channel *ec, FILE *file)

  Disconnect from a remote host and free the resources used.
- void rdma_recv (struct rdma_cm_id *id, struct ibv_mr *mr, FILE *file)

  A simple wrapper for rdma_post_recv()
- void rdma_send_op (struct rdma_cm_id *id, uint8_t op, FILE *file)

  A 0 byte send with immediate data.
- void rdma_write_inline (struct rdma_cm_id *id, void *buffer, uint64_t address, uint32_t key, FILE *file)

  A simple wrapper for an inline write using rdma_post_write().

4.2.1 Detailed Description

File containing the definitions of the functions listed in rdma_cs.h.

Author

Austin Pohlmann

4.2.2 Function Documentation

4.2.2.1 struct rdma_cm_id* cm_event (struct rdma_event_channel * *ec*, enum rdma_cm_event_type *expected*, FILE * *file*)

Process a communication manager event.

The function will call exit(-1) if the event found does not match the expected event

Returns

the `struct rdma_cm_id` of the new connection if the event was `RDMA_CM_EVENT_CONNECT_REQUEST`

Parameters

ec	the event channel to check
expected	the expected event
file	the file to output the connection info of a new connection if the event was `RDMA_CM_EVENT_CONNECT_REQUEST`

4.2.2.2 uint32_t get_completion (struct rdma_cm_id * *cm_id*, enum completion_type *type*, uint8_t *print*, FILE * *file*)

Wait for and pull a work completion.

This function will block until a work completion of the specified type is pulled from the completion queue. If the completion contains immediate data, it will be returned. Operations included in SEND: send (read and write if IBV_SEND_SIGNALED is set) Operations included in RECV: receive

Returns

the immediate data received, if present

Parameters

cm_id	the id associated with the connection to the remote host
type	the type of completion to pull, either SEND of RECV
print	1 if this should print anything, 0 if not
file	the file to print to

4.2.2.3 int obliterate (struct rdma_cm_id * *mine*, struct rdma_cm_id * *client*, struct ibv_mr * *mr*, struct rdma_event_channel * *ec*, FILE * *file*)

Disconnect from a remote host and free the resources used.

Parameters

mine	the id of the local host
client	the id of the remote host
mr	the memory region to free
ec	the event channel to use
file	the file to print to

4.2.2.4 void rdma_recv (struct rdma_cm_id * *id*, struct ibv_mr * *mr*, FILE * *file*)

A simple wrapper for rdma_post_recv()

Parameters

id	the id associated with the connection to the remote host
mr	the memory region to receive the data in
file	the file to print to in the event of an error

4.2.2.5 void rdma_send_op (struct rdma_cm_id * *id*, uint8_t *op*, FILE * *file*)

A 0 byte send with immediate data.

This is used to send opcodes between hosts using the immediate data.

Returns

NULL

Parameters

id	the id associated with the connection to the remote host
op	the opcode (immediate data) to send
file	the file to print to in the event of an error

4.2.2.6 void rdma_write_inline (struct rdma_cm_id * *id*, void * *buffer*, uint64_t *address*, uint32_t *key*, FILE * *file*)

A simple wrapper for an inline write using rdma_post_write().

Returns

NULL

Parameters

id	the id associated with the connection to the remote host
buffer	the buffer containing the data to be written
address	the remote address to write to
key	the key associated with the remote address
file	the file to print to in the event of an error

4.2.2.7 void stop_it (char * *reason*, int *error*, FILE * *file*)

Print an error message and exit the application.

Returns

NULL

Parameters

reason	a string representation of what the error came from
error	the error number, typically just errno
file	the file to output the error message to

4.2.2.8 void swap_info (struct rdma_cm_id * *cm_id*, struct ibv_mr * *mr*, uint32_t * *rkey*, uint64_t * *remote_addr*, size_t * *size*, FILE * *file*)

Exchange the information needed to perform rdma read/write operations.

The address, rkey, and size of a memory region is sent to and recieved from a remote host. WARNING: this erases the contents of the memory region used

Returns

NULL

Parameters

cm_id	the id associated with the connection to the remote host
mr	the memory region to send information about
rkey	the location to store the remote host's memory region's rkey
remote_addr	the location to store the remote host's memory region's address
size	the location to store the remote host's memory region's size
file	the file to print the sent and received information to

4.3 rdma_cs.h File Reference

The header file containing resources used in both the client and the server.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/time.h>
#include <unistd.h>
#include <semaphore.h>
#include <pthread.h>
#include <infiniband/verbs.h>
#include <infiniband/arch.h>
#include <rdma/rdma_cma.h>
#include <rdma/rdma_verbs.h>
```

Data Structures

- struct client

  Node for a linked lists containing information about open memory regions.

Macros

- #define MAX_SEND_WR 8

  The max amount of send type work requests.
- #define MAX_SEND_SGE 4

  The max amount of send type scatter/gather elements.
- #define MAX_RECV_WR 8

  The max amount of receive type work requests.
- #define MAX_RECV_SGE 4

  The max amount of receive type scatter/gather elements.
- #define MAX_INLINE_DATA 256

  The max amount (in bytes) that can be written inline.
- #define REGION_LENGTH 512

  The default local memory region size.
- #define SERVER_MR_SIZE 1024

  The default memory region size on the server.
- #define SERVER_LOG_PATH "./server_logs/"

  The file path to store the server logs to.

Enumerations

- enum completion_type { RECV, SEND }

  Determines the type of completion is being fetched.
- enum client_opcodes {
  DISCONNECT = 1, WRITE_INLINE, WRITE, READ,
  OPEN_MR, CLOSE_MR, ADD_CLIENT = 10, REMOVE_CLIENT }

  Standard opcodes for operations done between hosts.

Functions

- uint32_t get_completion (struct rdma_cm_id *, enum completion_type, uint8_t, FILE *)
  Wait for and pull a work completion.
- struct rdma_cm_id * cm_event (struct rdma_event_channel *, enum rdma_cm_event_type, FILE *)
  Process a communication manager event.
- void swap_info (struct rdma_cm_id *, struct ibv_mr *, uint32_t *, uint64_t *, size_t *, FILE *)
  Exchange the information needed to perform rdma read/write operations.
- int obliterate (struct rdma_cm_id *, struct rdma_cm_id *, struct ibv_mr *, struct rdma_event_channel *, FILE *)
  Disconnect from a remote host and free the resources used.
- void stop_it (char *, int, FILE *)
  Print an error message and exit the application.
- void rdma_recv (struct rdma_cm_id *, struct ibv_mr *, FILE *)
  A simple wrapper for rdma_post_recv()
- void rdma_send_op (struct rdma_cm_id *, uint8_t, FILE *)
  A 0 byte send with immediate data.
- void rdma_write_inline (struct rdma_cm_id *, void *, uint64_t, uint32_t, FILE *)
  A simple wrapper for an inline write using rdma_post_write().

4.3.1 Detailed Description

The header file containing resources used in both the client and the server.

Author

Austin Pohlmann

4.3.2 Enumeration Type Documentation

4.3.2.1 enum client_opcodes

Standard opcodes for operations done between hosts.

Enumerator

DISCONNECT Send a disconnect request to the remote host

WRITE_INLINE Perform an inline rdma write

WRITE Perform an rdma write

READ Perform and rdma read

OPEN_MR Open a memory region on the server

CLOSE_MR Close a memory region on the server

ADD_CLIENT Used to add an open memory regions to clients' lists

REMOVE_CLIENT Used to remove open memory regions from clients' lists

4.3.2.2 enum completion_type

Determines the type of completion is being fetched.

Enumerator

RECV Used for receive operations

SEND Used for send, rdma read, and rdma write operations (only for SIGNLAED rdma operations)

4.3.3 Function Documentation

4.3.3.1 struct rdma_cm_id* cm_event (struct rdma_event_channel * *ec*, enum rdma_cm_event_type *expected*, FILE * *file*)

Process a communication manager event.

The function will call exit(-1) if the event found does not match the expected event

Returns

the `struct rdma_cm_id` of the new connection if the event was `RDMA_CM_EVENT_CONNECT_REQUEST`

Parameters

ec	the event channel to check
expected	the expected event
file	the file to output the connection info of a new connection if the event was `RDMA_CM_EVENT_CONNECT_REQUEST`

4.3.3.2 uint32_t get_completion (struct rdma_cm_id * *cm_id*, enum completion_type *type*, uint8_t *print*, FILE * *file*)

Wait for and pull a work completion.

This function will block until a work completion of the specified type is pulled from the completion queue. If the completion contains immediate data, it will be returned. Operations included in SEND: send (read and write if IBV_SEND_SIGNALED is set) Operations included in RECV: receive

Returns

the immediate data received, if present

Parameters

cm_id	the id associated with the connection to the remote host
type	the type of completion to pull, either SEND of RECV
print	1 if this should print anything, 0 if not
file	the file to print to

4.3.3.3 int obliterate (struct rdma_cm_id * *mine*, struct rdma_cm_id * *client*, struct ibv_mr * *mr*, struct rdma_event_channel * *ec*, FILE * *file*)

Disconnect from a remote host and free the resources used.

Parameters

mine	the id of the local host
client	the id of the remote host
mr	the memory region to free
ec	the event channel to use
file	the file to print to

4.3.3.4 void rdma_recv (struct rdma_cm_id * *id*, struct ibv_mr * *mr*, FILE * *file*)

A simple wrapper for rdma_post_recv()

Parameters

id	the id associated with the connection to the remote host
mr	the memory region to receive the data in
file	the file to print to in the event of an error

4.3.3.5 void rdma_send_op (struct rdma_cm_id * *id*, uint8_t *op*, FILE * *file*)

A 0 byte send with immediate data.

This is used to send opcodes between hosts using the immediate data.

Returns

NULL

Parameters

id	the id associated with the connection to the remote host
op	the opcode (immediate data) to send
file	the file to print to in the event of an error

4.3.3.6 void rdma_write_inline (struct rdma_cm_id * *id*, void * *buffer*, uint64_t *address*, uint32_t *key*, FILE * *file*)

A simple wrapper for an inline write using rdma_post_write().

Returns

NULL

Parameters

id	the id associated with the connection to the remote host
buffer	the buffer containing the data to be written
address	the remote address to write to
key	the key associated with the remote address
file	the file to print to in the event of an error

4.3.3.7 void stop_it (char * *reason*, int *error*, FILE * *file*)

Print an error message and exit the application.

Returns

NULL

Parameters

reason	a string representation of what the error came from
error	the error number, typically just errno
file	the file to output the error message to

4.3.3.8 void swap_info (struct rdma_cm_id * *cm_id*, struct ibv_mr * *mr*, uint32_t * *rkey*, uint64_t * *remote_addr*, size_t * *size*, FILE * *file*)

Exchange the information needed to perform rdma read/write operations.

The address, rkey, and size of a memory region is sent to and recieved from a remote host. WARNING: this erases the contents of the memory region used

Returns

NULL

Parameters

cm_id	the id associated with the connection to the remote host
mr	the memory region to send information about
rkey	the location to store the remote host's memory region's rkey
remote_addr	the location to store the remote host's memory region's address
size	the location to store the remote host's memory region's size
file	the file to print the sent and received information to

4.4 server.c File Reference

A RDMA server This server uses pthreads for each client conection, as well as a listener thread and the main thread for server administration.

```
#include "rdma_cs.h"
```

Data Structures

- struct pnode

  Linked list node containing information on all running threads.
- struct cnode

  Linked list node containing information on all connected clients.

Enumerations

- enum client_status { OPEN, CLOSED }

  Determines if a client's memory region is open or closed to other clients.

Functions

- void binding_of_isaac (struct rdma_cm_id *cm_id, short port)

  Bind to a specified port.
- void * hey_listen (void *cmid)

  The funtion for the listener thread.
- void * secret_agent (void *id)

  The function for the agent threads.
- void add_thread (struct pnode node)

  Add a node to the thread list.
- void add_client (struct cnode node)

  Add a node to the client list.
- void remove_thread (pthread_t id)

  Remove a node from the thread list.
- void remove_client (pthread_t id)

  Remove a node from the client list.

- void set_status (pthread_t id, enum client_status status)

  Change the status of a client's memory region.
- void remote_remove (pthread_t id)

  Inform all clients when another client's memory region closes(but only if it was previously open).
- void remote_add (pthread_t id)

  Inform all clients when another client's memory region opens.
- int **main** (int argc, char ∗∗argv)

Variables

- struct pnode ∗ **tlist_head**
- struct cnode ∗ **clist_head**
- sem_t clist_sem

  Semaphore for synchronizing the manipulation of the client list.
- sem_t tlist_sem

  Semaphore for synchronizing the manipulation of the thread list.
- short port

  The port number that the server will be bound to.
- unsigned long clients = 0

  The current number of connected clients.
- unsigned long **idnum** = 0
- FILE ∗ log_p

  The file pointer of the log file.

4.4.1 Detailed Description

A RDMA server This server uses pthreads for each client conection, as well as a listener thread and the main thread for server administration.

Author

Austin Pohlmann

4.4.2 Enumeration Type Documentation

4.4.2.1 enum client_status

Determines if a client's memory region is open or closed to other clients.

Enumerator

OPEN Memory region is open for other clients to use

CLOSED Memory region is closed to other clients

4.4.3 Function Documentation

4.4.3.1 void add_client (struct cnode *node*)

Add a node to the client list.

Returns

NULL

Parameters

node	the node to add to the list

4.4.3.2 void add_thread (struct pnode *node*)

Add a node to the thread list.

Returns

`NULL`

Parameters

node	the node to add to the list

4.4.3.3 void binding_of_isaac (struct rdma_cm_id * *cm_id,* short *port*)

Bind to a specified port.

If `port` is 0, a random free port will be chosen

Returns

`NULL`

Parameters

cm_id	The server's communication manager id
port	the port to bind to

4.4.3.4 void * hey_listen (void * *cmid*)

The funtion for the listener thread.

Listens for incoming connection requests and passes the relevant information to the creation of a new secret_agent() thread.

Returns

`NULL`

Parameters

cmid	the `struct rdma_cm_id` of the server cast to be a `void *`

4.4.3.5 void remote_add (pthread_t *id*)

Inform all clients when another client's memory region opens.

Returns

`NULL`

Parameters

id	the thread ID of the client that opened their memory region

4.4.3.6 void remote_remove (pthread_t *id*)

Inform all clients when another client's memory region closes(but only if it was previously open).

Returns

NULL

Parameters

id	the thread ID of the client that closed their memory region

4.4.3.7 void remove_client (pthread_t *id*)

Remove a node from the client list.

Returns

NULL

Parameters

id	the thread id of the node to remove from the list

4.4.3.8 void remove_thread (pthread_t *id*)

Remove a node from the thread list.

Returns

NULL

Parameters

id	the id of the node to remove from the list

4.4.3.9 void * secret_agent (void * *id*)

The function for the agent threads.

Handles a single client connection

Returns

NULL

Parameters

id	the `struct rdma_cm_id` of the connection to the client cast to be a `void *`

4.4.3.10 void set_status (pthread_t *id,* enum **client_status** *status*)

Change the status of a client's memory region.

Returns

`NULL`

Parameters

id	the thread id of the client
status	the new status

Index