Implementation details

1. Create an OrderManager class derived from IORderManager.
2. Use the unordered_map container to store data for quick lookup. Created three core containers, m_order_submitted, m_order_activated, and m_order_cancelled.

    - m_order_submitted, which stores the order submitted by Trader.
    - m_order_activated [] stores the order confirmed by Exchange and uses the market number as the index of the array. Reduce the competition of resources by each thread, thus reducing the time to retrieve data.
    - m_order_cancelled, stores the canceled order.

3. Use read-write locks to increase the concurrency of data access.
4. The OrderIdentifier value is formed as a string in the format "_market%d_desk%d_trader%d_sequence%d" and used as a key for unordered_map for easy management and lookup.

Build & Run

Windows

Prepare

Create an empty project and copy the following files into Visual Studio 2017 (this is the version I am using).

Challenge.h

OrderManager.h

OrderManager.cpp

WriteFirstRWLock.h

TestMain.cpp

Compile & Run

Compile and run it directly.

Linux

Prepare

Create a directory and copy the following files.

challenge.h

OrderManager.h

OrderManager.cpp

WriteFirstRWLock.h

TestMain.cpp

makefile

Compile

Run the command in the project directory: make all

Note:

My test environment information is as follows:

g++: gcc version 7.3.0 (Ubuntu 7.3.0-16ubuntu3)

make: GNU Make 4.1

Run

Run the command in the project directory: make test > log_linux.txt

Host test

Case 01: Submit, Activate, and Transaction Order

Describe

1. Create three threads to implement order submission. creating thread 1 will submit 20 [0-19] orders, creating thread 2 will submit 10 [0-9] orders, creating thread 3 will submit 5 [0-4] orders.
2. Create two threads to activate the order. The first thread will activate 10 orders created by creating thread 1[0-4] and creating thread 2[0-4]. The second thread will submit 9 orders created by creating thread 2 [5-9] and creating thread 3 [0-3].
3. Create two threads to trade the order. The first thread will trade 4 orders created by creating threads 1 [0-1] and 2 [0-1]. The second thread will trade 11 orders created by creating thread 1 [2-3], creating thread 2 [2-7] and creating thread 3 [0-2].

Results

==========

order_submitted

_market0_desk0_trader0_sequence19 => (p:10, q:12)
_market0_desk0_trader0_sequence18 => (p:10, q:12)
_market0_desk0_trader0_sequence17 => (p:10, q:12)
_market0_desk0_trader0_sequence16 => (p:10, q:12)
_market0_desk0_trader0_sequence15 => (p:10, q:12)
_market0_desk0_trader0_sequence10 => (p:10, q:12)
_market0_desk0_trader0_sequence5 => (p:10, q:12)
_market0_desk0_trader0_sequence7 => (p:10, q:12)

_market0_desk0_trader0_sequence8 => (p:10, q:12)
_market0_desk0_trader0_sequence6 => (p:10, q:12)
_market0_desk0_trader0_sequence9 => (p:10, q:12)
_market0_desk0_trader0_sequence14 => (p:10, q:12)
_market2_desk0_trader2_sequence4 => (p:10, q:12)
_market0_desk0_trader0_sequence11 => (p:10, q:12)
_market0_desk0_trader0_sequence12 => (p:10, q:12)
_market0_desk0_trader0_sequence13 => (p:10, q:12)
==========

order_activated

_market0_desk0_trader0_sequence4 => (p:10, q:12)
_market1_desk0_trader1_sequence9 => (p:10, q:12)
_market1_desk0_trader1_sequence8 => (p:10, q:12)
_market2_desk0_trader2_sequence3 => (p:10, q:12)
==========

order_cancelled

_market2_desk0_trader2_sequence1 => (p:10, q:0)
_market2_desk0_trader2_sequence0 => (p:10, q:0)
_market1_desk0_trader1_sequence7 => (p:10, q:0)
_market1_desk0_trader1_sequence5 => (p:10, q:0)
_market1_desk0_trader1_sequence6 => (p:10, q:0)
_market1_desk0_trader1_sequence4 => (p:10, q:0)
_market1_desk0_trader1_sequence3 => (p:10, q:0)
_market1_desk0_trader1_sequence2 => (p:10, q:0)
_market0_desk0_trader0_sequence1 => (p:10, q:0)
_market0_desk0_trader0_sequence0 => (p:10, q:0)
_market2_desk0_trader2_sequence2 => (p:10, q:0)
_market0_desk0_trader0_sequence2 => (p:10, q:0)
_market1_desk0_trader1_sequence0 => (p:10, q:0)
_market0_desk0_trader0_sequence3 => (p:10, q:0)

_market1_desk0_trader1_sequence1 => (p:10, q:0)

Case 02: Submit, activate and cancel orders

Describe

1. Submit and activate the orders only, as in case 01.
2. The trader will cancel the order created by creating thread 1 [0-1] and creating thread 2 [0-2].
3. Get the order quantity, they are still valid. Because Exchange has not yet replied to cancel confirmation.
4. The exchange will cancel the order created by creating thread 1 [0], creating thread 2 [0-1], and creating thread 3 [0-2].
5. Get the order quantity again, at this time they should be canceled.

Results

(M:0,D:0,T:0,I:0)=>0

... ...

(M:1,D:0,T:1,I:0)=>0

(M:1,D:0,T:1,I:1)=>0

... ...

(M:2,D:0,T:2,I:0)=>0

(M:2,D:0,T:2,I:1)=>0

(M:2,D:0,T:2,I:2)=>0

(M:2,D:0,T:2,I:3)=>12

(M:2,D:0,T:2,I:4)=>0

==========

order_submitted

_market2_desk0_trader2_sequence4 => (p:10, q:12)

_market0_desk0_trader0_sequence19 => (p:10, q:12)

_market0_desk0_trader0_sequence18 => (p:10, q:12)
_market0_desk0_trader0_sequence17 => (p:10, q:12)
_market0_desk0_trader0_sequence16 => (p:10, q:12)
_market0_desk0_trader0_sequence14 => (p:10, q:12)
_market0_desk0_trader0_sequence13 => (p:10, q:12)
_market0_desk0_trader0_sequence12 => (p:10, q:12)
_market0_desk0_trader0_sequence11 => (p:10, q:12)
_market0_desk0_trader0_sequence15 => (p:10, q:12)
_market0_desk0_trader0_sequence10 => (p:10, q:12)
_market0_desk0_trader0_sequence7 => (p:10, q:12)
_market0_desk0_trader0_sequence9 => (p:10, q:12)
_market0_desk0_trader0_sequence6 => (p:10, q:12)
_market0_desk0_trader0_sequence5 => (p:10, q:12)
_market0_desk0_trader0_sequence8 => (p:10, q:12)
==========

order_activated

_market0_desk0_trader0_sequence4 => (p:10, q:12)
_market0_desk0_trader0_sequence3 => (p:10, q:12)
_market0_desk0_trader0_sequence2 => (p:10, q:12)
_market0_desk0_trader0_sequence1 => (p:10, q:12)
_market1_desk0_trader1_sequence9 => (p:10, q:12)
_market1_desk0_trader1_sequence8 => (p:10, q:12)
_market1_desk0_trader1_sequence7 => (p:10, q:12)
_market1_desk0_trader1_sequence5 => (p:10, q:12)
_market1_desk0_trader1_sequence2 => (p:10, q:12)
_market1_desk0_trader1_sequence4 => (p:10, q:12)
_market1_desk0_trader1_sequence6 => (p:10, q:12)
_market1_desk0_trader1_sequence3 => (p:10, q:12)
_market2_desk0_trader2_sequence3 => (p:10, q:12)
==========

order_cancelled

_market2_desk0_trader2_sequence2 => (p:10, q:12)

_market2_desk0_trader2_sequence1 => (p:10, q:12)

_market2_desk0_trader2_sequence0 => (p:10, q:12)

_market1_desk0_trader1_sequence1 => (p:10, q:12)

_market1_desk0_trader1_sequence0 => (p:10, q:12)

_market0_desk0_trader0_sequence0 => (p:10, q:12)