Group By implementation not thread safe #49

New issue

Closed **mattpodwysocki** opened this issue on 14 Oct 2014 · 3 comments

mattpodwysocki commented on 14 Oct 2014 Contributor

Copied from https://rx.codeplex.com/workitem/40

Looks like the group by implementation is not thread safe. Internally it uses a Dictionary.Add, in the onnext of the Sink, without using proper locking mechanisms.

```
if (!_map.TryGetValue(key, out writer)) {
    writer = new Subject<TElement>();
    _map.Add(key, writer);
   fireNewMapEntry = true;
}
```

I noticed by subscribing to an event pattern that is invoked by multiple threads at the same time. (In my case 24 threads that potentially call TransportMessageReceived concurrently)

```
Observable.FromEventPattern<TransportMessageReceivedEventArgs>(
                    h => _bus.Transport.TransportMessageReceived += h,
                    h => _bus.Transport.TransportMessageReceived -= h);
```

And simply subscribe to it with a group by expression

```
var messagesReceived = from e in observable
                       group e by "something" into c
                       select c ;

subscription = messagesReceived.Subscribe(r => Console.WriteLine("Triggered") );
```

This will occasionally throw a null reference in `Dictionary.Insert` ... as that implementation is not thread safe by default.

1

mattpodwysocki commented on 14 Oct 2014 Contributor Author

davedev wrote Jul 3, 2013 at 7:50 PM

Hi,

That is the correct behavior.

One of Rx's contracts is that notifications must be pushed serially; i.e., you must not invoke OnNext concurrently within a single observable. There are probably many Rx operators that may cause threading bugs if you do not satisfy this contract.

See §4.2 in the Rx Design Guidelines document.
▪Dave

mattpodwysocki commented on 14 Oct 2014 Contributor Author

Bnaya wrote Aug 10, 2013 at 3:15 PM

Isn't it time to re-consider this limitation?
it is true that this is the current contract, but many operatoir can be thread-safe with a little effort.
I believe that it should change, maybe by adding a set of thread-safe operators.
without doing it there is a need for synchronization and we may not be able to get full utilization of a multi core environment (see Amdahl's law).

mattpodwysocki commented on 14 Oct 2014 Contributor Author

davedev wrote Aug 10, 2013 at 6:54 PM

I reject the premise. A thread-safe operator is thread safe because it ensures synchronization, so whether Rx does it or you do it yourself (e.g., via the Synchronize operator) it's going to end up being the same thing anyway. A set of thread-safe operators would essentially be, perhaps, identical to the set of existing operators with the addition of the Synchronize operator.

mattpodwysocki closed this on 14 Oct 2014

akarnokd mentioned this issue on 23 Nov 2019

Deadlock on OSX #1092 Closed

Sign up for free **to join this conversation on GitHub**. Already have an account? Sign in to comment

Assignees
No one assigned

Labels
None yet

Projects
None yet

Milestone
No milestone

Linked pull requests
Successfully merging a pull request may close this issue.

1 participant