Ruby - Feature #8128

New primitives for Rinda::TupleSpace

03/20/2013 08:00 AM - vjoel (Joel VanderWerf)

Status:	Rejected	
Priority:	Normal	
Assignee:	seki (Masatoshi Seki)	
Target version:		

Description

=begin

= New primitives for Rinda::TupleSpace

This issue proposes adding two new primitives to TupleSpace for atomic bulk operations:

== 1. TupleSpace#replace_all

=== What it does

Calling

replace_all(tuple, new_tuple, sec=nil)

atomically removes all tuples matching ((|tuple|)) and writes ((|new_tuple|)). It does not block waiting for tuples. The return value is a pair:

[matching_tuples, entry]

where ((|matching_tuples|)) is like the return value of (({read_all(tuple)})) and
((|entry|)) is like the return value of (({write(new_tuple)})).

=== Why it is needed

It is not possible to do this atomically with existing primitives. As noted in ((*The dRuby Book*)), p. 176, "It isn't easy to represent a dictionary using TupleSpace." Essentially, the #[]= and #[] operations must take/write a global lock tuple.

Using #replace_all, it is easy to implement a key-value store without lock tuples. See key-value-store.rb for an example.

=== Modularity

The new code is entirely contained in two modules in a single separate file. These modules are included/extended to TupleSpace and TupleSpaceProxy as desired to add the replace_all functionality.

=== Examples

See key-value-store.rb and example-replace-all.rb.

== 2. TupleSpace#take_all

=== What it does

Calling

take_all(tuple)

atomically removes all matching tuples. It does not block waiting for tuples. The return value is the array of tuples, like the return value of (({read_all(tuple)})).

=== Why it is needed

It is not possible to do this atomically with existing primitives, though in this case atomicity may not be important. More importantly, it is not possible to do this efficiently with existing primitives. The best approximation would be an unbounded sequence of #take calls.

=== Modularity

The new code is entirely contained in two modules in a single separate file. These modules are included/extended to TupleSpace and TupleSpaceProxy as desired to add the take_all functionality.

=== Examples

See example-take-all.rb.

=end

History

#1 - 03/20/2013 08:06 AM - hsbt (Hiroshi SHIBATA)

- Assignee set to seki (Masatoshi Seki)

#2 - 03/23/2013 04:45 PM - drbrain (Eric Hodel)

- Status changed from Open to Assigned

- Priority changed from 3 to Normal

A TupleSpace isn't designed for these types of operations, see: http://www.lindaspaces.com/book/ (the TupleSpace book).

#3 - 03/24/2013 06:18 AM - seki (Masatoshi Seki)

- Status changed from Assigned to Rejected

I think so: https://twitter.com/drbrain/status/315510564233293825

This is a global lock. If you want a KVS, I recommend the Hash or Drip.

#4 - 03/27/2013 08:06 AM - vjoel (Joel VanderWerf)

You are right: it is best to leave these extensions out of trunk. I can maintain them separately.

However, some clarifications:

- the proposed #take_all and #replace_all operations do *not* block. They are like #read_all in that respect.
- #read_all is not one of the original Linda primitives, either, AFAICT. (Yet, I cannot imagine using linda/rinda without it :)

Thanks for all your excellent work on distributed ruby, Eric and Masatoshi. Cheers!

Files

replace-all.rb	1.59 KB	03/20/2013	vjoel (Joel VanderWerf)
take-all.rb	961 Bytes	03/20/2013	vjoel (Joel VanderWerf)
example-replace-all.rb	910 Bytes	03/20/2013	vjoel (Joel VanderWerf)
example-take-all.rb	813 Bytes	03/20/2013	vjoel (Joel VanderWerf)
key-value-store.rb	1.13 KB	03/20/2013	vjoel (Joel VanderWerf)