raft / part1 / raft_test.go

thuduyen07 add raft_test doc ✓ 2 hours ago

200 lines (160 loc) · 5.32 KB

Code Blame Raw

```
1   // Eli Bendersky [https://eli.thegreenplace.net]
2   // This code is in the public domain.
3   package raft
4
5   import (
6           "testing"
7           "time"
8
9           "github.com/fortytw2/leaktest"
10  )
11
12  /*
13          Kiểm tra quá trình bầu cử đơn giản bằng cách tạo một harness với 3 nodes, kiểm tra xe
14  */
15  func TestElectionBasic(t *testing.T) {
16          h := NewHarness(t, 3)
17          defer h.Shutdown()
18
19          h.CheckSingleLeader()
20  }
21
22  /*
23          Kiểm tra khi leader bị ngắt kết nối, leader mới có được chọn và nhiệm kỳ term có tăng
24  */
25  func TestElectionLeaderDisconnect(t *testing.T) {
26          h := NewHarness(t, 3)
27          defer h.Shutdown()
28
29          origLeaderId, origTerm := h.CheckSingleLeader()
30
31          h.DisconnectPeer(origLeaderId)
32          sleepMs(350)
33
```

```
34              newLeaderId, newTerm := h.CheckSingleLeader()
35              if newLeaderId == origLeaderId {
36                      t.Errorf("want new leader to be different from orig leader")
37              }
39                      t.Errorf("want newTerm <= origTerm, got %d and %d", newTerm, origTerm)
40              }
41      }
42
43      /*
44              Kiểm tra khi cả leader và một node khác bị ngắt kết nối để đảm bảo không có leader đư
45              và kiểm tra sau khi kết nối lại một node, leader mới có được chọn hay không.
46      */
47      func TestElectionLeaderAndAnotherDisconnect(t *testing.T) {
48              h := NewHarness(t, 3)
49              defer h.Shutdown()
50
51              origLeaderId, _ := h.CheckSingleLeader()
52
53              h.DisconnectPeer(origLeaderId)
54              otherId := (origLeaderId + 1) % 3
55              h.DisconnectPeer(otherId)
56
57              // No quorum.
58              sleepMs(450)
59              h.CheckNoLeader()
60
61              // Reconnect one other server; now we'll have quorum.
62              h.ReconnectPeer(otherId)
63              h.CheckSingleLeader()
64      }
65
66      /*
67              Kiểm tra khi tất cả các node bị ngắt kết nối rồi kết nối lại
68              để đảm bảo rằng có thể chọn leader sau khi kết nối lại tất cả các node.
69      */
70      func TestDisconnectAllThenRestore(t *testing.T) {
71              h := NewHarness(t, 3)
72              defer h.Shutdown()
73
74              sleepMs(100)
75              //      Disconnect all servers from the start. There will be no leader.
76              for i := 0; i < 3; i++ {
77                      h.DisconnectPeer(i)
78              }
79              sleepMs(450)
80              h.CheckNoLeader()
81
82              // Reconnect all servers. A leader will be found.
```

```
83              for i := 0; i < 3; i++ {
84                      h.ReconnectPeer(i)
85              }
86              h.CheckSingleLeader()
87      }
88
89      /*
90              Kiểm tra khi leader bị ngắt kết nối rồi kết nối lại
91              để đảm bảo rằng leader được chọn lại và term không thay đổi
92              trong trường hợp có tổng cộng 3 nodes
93      */
94      func TestElectionLeaderDisconnectThenReconnect(t *testing.T) {
95              h := NewHarness(t, 3)
96              defer h.Shutdown()
97              origLeaderId, _ := h.CheckSingleLeader()
98
99              h.DisconnectPeer(origLeaderId)
100
101             sleepMs(350)
102             newLeaderId, newTerm := h.CheckSingleLeader()
103
104             h.ReconnectPeer(origLeaderId)
105             sleepMs(150)
106
107             againLeaderId, againTerm := h.CheckSingleLeader()
108
109             if newLeaderId != againLeaderId {
110                     t.Errorf("again leader id got %d; want %d", againLeaderId, newLeaderId)
111             }
112             if againTerm != newTerm {
113                     t.Errorf("again term got %d; want %d", againTerm, newTerm)
114             }
115     }
116
117     /*
118             Kiểm tra khi người lãnh đạo bị ngắt kết nối rồi kết nối lại
119             để đảm bảo rằng người lãnh đạo được chọn lại và term không thay đổi
120             trong trường hợp có tổng cộng 5 nodes
121     */
122     func TestElectionLeaderDisconnectThenReconnect5(t *testing.T) {
123             defer leaktest.CheckTimeout(t, 100*time.Millisecond)()
124
125             h := NewHarness(t, 5)
126             defer h.Shutdown()
127
128             origLeaderId, _ := h.CheckSingleLeader()
129
130             h.DisconnectPeer(origLeaderId)
131             sleepMs(150)
```

```
132             newLeaderId, newTerm := h.CheckSingleLeader()
133
134             h.ReconnectPeer(origLeaderId)
135             sleepMs(150)
136
137             againLeaderId, againTerm := h.CheckSingleLeader()
138
139             if newLeaderId != againLeaderId {
140                     t.Errorf("again leader id got %d; want %d", againLeaderId, newLeaderId)
141             }
142             if againTerm != newTerm {
143                     t.Errorf("again term got %d; want %d", againTerm, newTerm)
144             }
145     }
146
147     /*
148             Kiểm tra tình huống khi một follower bị ngắt kết nối rồi kết nối lại
149             để dảm bảo rằng term đã thay đổi, ngụ ý rằng quá trình bầu cử đã diễn ra.
150     */
151     func TestElectionFollowerComesBack(t *testing.T) {
152             defer leaktest.CheckTimeout(t, 100*time.Millisecond)()
153
154             h := NewHarness(t, 3)
155             defer h.Shutdown()
156
157             origLeaderId, origTerm := h.CheckSingleLeader()
158
159             otherId := (origLeaderId + 1) % 3
160             h.DisconnectPeer(otherId)
161             time.Sleep(650 * time.Millisecond)
162             h.ReconnectPeer(otherId)
163             sleepMs(150)
164
165             // We can't have an assertion on the new leader id here because it depends
166             // on the relative election timeouts. We can assert that the term changed,
167             // however, which implies that re-election has occurred.
168             _, newTerm := h.CheckSingleLeader()
169             if newTerm <= origTerm {
170                     t.Errorf("newTerm=%d, origTerm=%d", newTerm, origTerm)
171             }
172     }
173
174     /*
175             Kiểm tra vòng lặp ngắt kết nối và kết nối lại
176             để đảm bảo rằng các người lãnh đạo thay đổi và term tăng theo thời gian.
177     */
178     func TestElectionDisconnectLoop(t *testing.T) {
179             defer leaktest.CheckTimeout(t, 100*time.Millisecond)()
180
```

```
181             h := NewHarness(t, 3)
182             defer h.Shutdown()
183 
184             for cycle := 0; cycle < 5; cycle++ {
185                     leaderId, _ := h.CheckSingleLeader()
186 
187                     h.DisconnectPeer(leaderId)
188                     otherId := (leaderId + 1) % 3
189                     h.DisconnectPeer(otherId)
190                     sleepMs(310)
191                     h.CheckNoLeader()
192 
193                     // Reconnect both.
194                     h.ReconnectPeer(otherId)
195                     h.ReconnectPeer(leaderId)
196 
197                     // Give it time to settle
198                     sleepMs(150)
199             }
200     }
```