Passive Application Monitoring with LTM

Deb Allen, 2008-04-06

LTM has a broad variety of application monitoring tools you can use to address almost any high availability application monitoring requirement. This article will explain how to configure basic passive application monitoring in LTM 9.2.0 through 9.4.x.

Passive Monitoring Overview

A standard active health monitor initiates a request to each pool member to verify its health. This is generally the most reliable way of determining the health of the server with regard to the load balanced service, but may impose excessive overhead on busy servers when run at the frequent intervals required to prevent sending clients to recently failed servers.

An alternative approach to active health monitoring is to have LTM passively monitor the server side of all load balanced requests for refused connections, application errors, abnormal session terminations, etc., and disable the server if excessive connection- or application-related problems are seen.

HTTP Profile Option

The http profile contains an option to passively monitor server responses and redirect specific connections to an alternate URL based on error responses seen on that same connection. This solution is limited to HTTP applications, based on HTTP response codes only, and the remediation is limited to a redirect of the current connection. It does not affect the server status, which means that other connections are not prevented from being load balanced to a server from which questionable responses are seen across multiple connections.

iRule + Monitor Option

For more granular control, you can combine an iRule and an application health monitor. The iRule can observe and analyze any part of a server response and correlate response patterns across multiple connections to determine the health of a pool member. The iRule can also include logic to respond to excessive errors by redirecting, reselecting, or sending a custom response, and by disabling the pool member. The health monitor will continue to monitor disabled pool members and re-enable them when they are again responding as expected.

For this solution, you will:

1. Configure your virtual server and pool with the settings and profiles appropriate for your application
2. Create and apply an iRule that monitors server response traffic and disables pool members based on observed error conditions
3. Create and apply an application health monitor to re-instate recovered pool members previously disabled

The iRule

The iRule will use the built-in response events to analyze server responses and use the LB::down command (introduced in LTM 9.2.0) to disable apparently unhealthy pool members. The iRule may consider response codes, headers, errors, intermittent lack of response, or any specific data contained in the response. In most cases it will make sense to enforce a threshold error rate per pool member. A number of different algorithms may be used to determine the conditions under which a pool member should be disabled.

For example, this HTTP iRule was created to disable a pool member on a server error response (50x) when more than 2 errors are seen within the same second:

```
when RULE_INIT {
  set ::MaxErrSec 2
}
```

```
when HTTP_RESPONSE {
  if { [HTTP::status] starts_with “50” } {
    # init counter variable if it doesn’t exist yet
    if { [catch {eval $::Counter_[LB::server]}] } {
      set ::Counter_[LB::server] 0
    }
    # init timer variable if it doesn’t exist yet
    if { [catch {eval $::LastErr_[LB::server]}] } {
      set ::LastErr_[LB::server] 0
    }
    # if still in same second
    if { [clock seconds] == $::LastErr_[LB::server] } {
      # increment counter
      incr ::Counter_[LB::server]
      # and disable pool member if error rate is exceeded
      if { $::Counter_[LB::server] > $::MaxErrSec }{
        LB::down
      }
    } else {
      # otherwise reset timer and counter
      set ::LastErr_[LB::server] [clock seconds]
      set ::Counter_[LB::server] 1
    }
  }
}
```

A number of other approaches could be used to accomplish the same objectives. For example, you could use more complex rate calculation logic as that seen in Kirk Bauer’s HTTP request throttling iRule in the codeshare: http://devcentral.f5.com/wiki/default.aspx/iRules/HTTPRequestThrottle.html

The Monitor

You will then create a standard application service monitor that verifies that the application is behaving as expected to a specific request. Use all the same settings in configuring it as you would for active monitoring except for the timing.

Since the sole function of this monitor is to resurrect disabled pool members once they again appear healthy, the interval will be long: At least as long as the rate interval used by the iRule (in this case, 1 minute).

Timeout is not applicable to this solution, but must be configured in the monitor template: Set the Timeout to a value at least 1 second longer than the interval. (The 3* + 1 guideline doesn’t apply in this case since the interval is much longer than a typical transaction timeout.)

Caveats

The timing for re-enabling pool member is less than deterministic, since it depends on when in the monitor interval the pool member was disabled by the iRule: A pool member may be marked down at any point, but cannot be marked up again until the monitor runs again, so the pool member may be down for almost an entire interval. Some experimentation with the interval length may be necessary to find the right balance between avoiding failing servers and leaving recovered servers idle longer than necessary.

Flapping (servers being marked down then back up again in rapid succession) may occur if the conditions tested by the monitor and the iRule are not convergent. For example, if your iRule is sees a specific server error and marks a pool member down because it has intermittent database connectivity, but your health monitor is just requesting a static HTML page, the health monitor may continue to mark the pool member up when it is actually unable to service requests. If pool member flapping is observed, adjust the health monitor to make a request more likely to reflect the application issues to which the iRule is intended to react.

Note: This article (specifically the LB::down approach) applies to LTM 9.2.0 through 9.4.x. Look for enhancements to LTM's passive application monitoring capabilities in upcoming releases.

Get the Flash Player to see this player.

20080604-LTMPassiveMonitoring.mp3

F5 Networks, Inc. | 401 Elliot Avenue West, Seattle, WA 98119 | 888-882-4447 | f5.com

F5 Networks, Inc.	F5 Networks	F5 Networks Ltd.	F5 Networks
Corporate Headquarters	Asia-Pacific	Europe/Middle-East/Africa	Japan K.K.
info@f5.com	apacinfo@f5.com	emeainfo@f5.com	f5j-info@f5.com

©2016 F5 Networks, Inc. All rights reserved. F5, F5 Networks, and the F5 logo are trademarks of F5 Networks, Inc. in the U.S. and in certain other countries. Other F5 trademarks are identified at f5.com. Any other products, services, or company names referenced herein may be trademarks of their respective owners with no endorsement or affiliation, express or implied, claimed by F5. CS04-00015 0113