iRules Optimization 101 - #05 - Evaluating iRule Performance

Deb Allen, 2007-07-11

Customers frequently ask, as poster CodeIT did:

> "I am wondering what the effect of writing more elaborate iRules will have on the F5's memory and processor.
> Is there a way to predict the effect on the F5's resources for a given rule and traffic load?"

In this article, I'll show you how to collect and interpret iRule runtime statistics to determine whether one version of a rule is more efficient than another, or to estimate a theoretical maximum number of connections possible for an iRule running on your LTM. Other articles in the series:

- iRules Optimization 101 – #01 – If, Elseif, and Switch
- iRules Optimization 101 – #02 – Expressions and Variables
- iRules Optimization 101 – #03 – For vs Foreach
- iRules Optimization 101 – #04 – Delimiters: Braces, Brackets, Quotes, and more
- iRules Optimization 101 – #05 – Evaluating iRule Performance

(Props to unRuleY, a1l0s2k9, citizen_elah, Joe, acarandang, and stephengun for insightful posts on this topic.)

Collecting Statistics

To generate & collect runtime statistics, you can insert the command "timing on" into your iRule. When you run traffic through your iRule with timing enabled, LTM will keep track of how many CPU cycles are spent evaluating each iRule event. You can enable rule timing for the entire iRule, or only for specific events.

To enable timing for the entire iRule, insert the "timing on" command at the top of the rule before the first "when EVENT_NAME" clause.
Here's an example enabling timing for all events in a rule:

```
rule my_fast_rule { timing on when HTTP_REQUEST { # Do some stuff } }
```

To enable iRule timing for only a specific event, insert the "timing on" command between the "when EVENT_NAME" declaration and the open curly brace.
Here's an example enabling timing for only a specific event:

```
rule my_slow_rule { when HTTP_REQUEST timing on { # Do some other stuff } }
```

(See the wiki page for the timing command for more details on timing only selected events by toggling timing on & off.)

With the timing command in place, each time the rule is evaluated, LTM will collect the timing information for the requested events.

To get a decent average for each of the events, you'll want to run at least a couple thousand iterations of the iRule under the anticipated production load.

Viewing Statistics

The statistics for your iRule (as measured in CPU cycles) may be viewed at the command line or console by running

```
bigpipe rule rule_name show all
```

The output includes totals for executions, failures & aborts along with minimum, average & maximum cycles consumed for each event since stats were last cleared.

```
RULE rule_name +-> HTTP_REQUEST 729 total 0 fail 0 abort | Cycles (min, avg, max) = (3693, 3959, 539
```

If you use the iRules Editor, you can instead view the same stats on the Statistics tab of the Properties dialog. (Double-click on the iRule in the left pane to open the Properties dialog.)

Evaluating statistics

Average cycles reported is the most useful metric of real-world performance, assuming a large representative load sample was evaluated.

The maximum cycles reported is often very large since it includes some one-time and periodic system overhead. (More on that below.)

Here's a spreadsheet (iRules Runtime Calculator) that will calculate percentage of CPU load per iteration once you populate it with your clock speed and the statistics gathered with the "timing" command. (Clock speed can be found by running 'cat /proc/cpuinfo' at the command line.)

Caveats

Timing is intended to be used only as an optimization/debug tool, and does have a small impact on performance, so don't leave it turned on indefinitely.

Timing functionality seems to exhibit a 70 - 100 cycle margin of error.

Use average cycles for most analyses. Maximum cycles is not always an accurate indicator of actual iRule performance, as the very first call a newly edited iRule includes the cycles consumed for compile-time optimizations, which will be reflected in an inflated maximum cycles value. The simple solution to this is to wait until the first time the rule is hit, then reset the statistics.

To reset statistics at the command line:

```
bigpipe rule [rule_name | all] stats reset
```

or "Reset Statistics" in the Statistics tab of the iRules Editor.

However, maximum cycles is also somewhat inflated by OS scheduling overhead incurred at least once per tick, so the max value is often overstated even if stats are cleared after compilation.

Links

Wiki page for timing command
iRules Runtime Calculator

Get the Flash Player to see this player.

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

F5 Networks, Inc.
Corporate Headquarters
info@f5.com

F5 Networks
Asia-Pacific
apacinfo@f5.com

F5 Networks Ltd.
Europe/Middle-East/Africa
emeainfo@f5.com

F5 Networks
Japan K.K.
f5j-info@f5.com