Linux bcc/BPF Run Queue (Scheduler) Latency

08 Oct 2016

I added this program to bcc earlier this year, and wanted to summarize it here as it fulfills an important need: examining scheduler run queue latency. It may not actually be a queue these days, but this metric has been called "run queue latency" for years: it's measuring the time from when a thread becomes runnable (eg, receives an interrupt, prompting it to process more work), to when it actually begins running on a CPU. Under CPU saturation, you can imagine threads have to wait their turn. But it can also happen for other weird scenarios, and there are cases where it can be tuned and reduced, improving overall system performance.

The program is runqlat, and it summarizes run queue latency as a histogram. Here is a heavily loaded system:

```
# runqlat
Tracing run queue latency... Hit Ctrl-C to end.
^C
     usecs               : count     distribution
         0 -> 1          : 233      |***********                             |
         2 -> 3          : 742      |************************************    |
         4 -> 7          : 203      |**********                              |
         8 -> 15         : 173      |********                                |
        16 -> 31         : 24       |*                                       |
        32 -> 63         : 0        |                                        |
        64 -> 127        : 30       |*                                       |
       128 -> 255        : 6        |                                        |
       256 -> 511        : 3        |                                        |
       512 -> 1023       : 5        |                                        |
      1024 -> 2047       : 27       |*                                       |
      2048 -> 4095       : 30       |*                                       |
      4096 -> 8191       : 20       |                                        |
      8192 -> 16383      : 29       |*                                       |
     16384 -> 32767      : 809      |****************************************|
     32768 -> 65535      : 64       |***                                     |
```

The distribution is bimodal, with one mode between 0 and 15 microseconds, and another between 16 and 65 milliseconds. These modes are visible as the spikes in the ASCII distribution (which is merely a visual representation of the "count" column). As an example of reading one line: 809 events fell into the 16384 to 32767 microsecond range (16 to 32 ms) while tracing.

A -m option can be used to show milliseconds instead, as well as an interval and a count. For example, showing three x five second summary in milliseconds:

```
# runqlat -m 5 3
Tracing run queue latency... Hit Ctrl-C to end.

     msecs               : count     distribution
         0 -> 1          : 3818     |****************************************|
         2 -> 3          : 39       |                                        |
         4 -> 7          : 39       |                                        |
         8 -> 15         : 62       |                                        |
        16 -> 31         : 2214     |***********************                 |
        32 -> 63         : 226      |**                                      |

     msecs               : count     distribution
         0 -> 1          : 3775     |****************************************|
         2 -> 3          : 52       |                                        |
         4 -> 7          : 37       |                                        |
         8 -> 15         : 65       |                                        |
        16 -> 31         : 2230     |***********************                 |
        32 -> 63         : 212      |**                                      |

     msecs               : count     distribution
         0 -> 1          : 3816     |****************************************|
         2 -> 3          : 49       |                                        |
         4 -> 7          : 40       |                                        |
         8 -> 15         : 53       |                                        |
        16 -> 31         : 2228     |***********************                 |
        32 -> 63         : 221      |**                                      |
```

This shows a similar distribution across the three summaries.

Here is the same system, but when it is CPU idle:

```
# runqlat 5 1
Tracing run queue latency... Hit Ctrl-C to end.

     usecs               : count     distribution
         0 -> 1          : 2250     |********************************        |
         2 -> 3          : 2340     |*********************************       |
         4 -> 7          : 2746     |****************************************|
         8 -> 15         : 418      |******                                  |
        16 -> 31         : 93       |*                                       |
        32 -> 63         : 28       |                                        |
        64 -> 127        : 119      |*                                       |
       128 -> 255        : 9        |                                        |
       256 -> 511        : 4        |                                        |
       512 -> 1023       : 20       |                                        |
      1024 -> 2047       : 22       |                                        |
      2048 -> 4095       : 5        |                                        |
      4096 -> 8191       : 2        |                                        |
```

Back to a microsecond scale, this time there is little run queue latency past 1 millisecond, as would be expected.

This tool can filter for specific pids or tids. Here is the USAGE message:

```
# runqlat -h
usage: runqlat [-h] [-T] [-m] [-P] [-L] [-p PID] [interval] [count]

Summarize run queue (schedular) latency as a histogram

positional arguments:
  interval            output interval, in seconds
  count               number of outputs

optional arguments:
  -h, --help          show this help message and exit
  -T, --timestamp     include timestamp on output
  -m, --milliseconds  millisecond histogram
  -P, --pids          print a histogram per process ID
  -L, --tids          print a histogram per thread ID
  -p PID, --pid PID   trace this PID only

examples:
    ./runqlat            # summarize run queue latency as a histogram
    ./runqlat 1 10       # print 1 second summaries, 10 times
    ./runqlat -mT 1      # 1s summaries, milliseconds, and timestamps
    ./runqlat -P         # show each PID separately
    ./runqlat -p 185     # trace PID 185 only
```

Also in bcc is cpudist, written by Sasha Goldshtein, which shows the time threads spent running on-CPU, rather than the time waiting for a turn.

You can comment here, but I can't guarantee your comment will remain here forever: I might switch comment systems at some point (eg, if disqus add advertisements).