Linux iosnoop Latency Heat Maps

23 Jul 2014

Odd patterns of I/O latency can be hidden by line graphs and summary statistics, and revealed by histograms and heat maps. In my previous post I showed my Linux iosnoop tool, which can trace block device I/O along with timestamps and latency. This information can be visualized, revealing any odd patterns.

As an example, I'll make a latency heat map from iosnoop output using my trace2heatmap.pl program (which is also on github). Here's the command I used, which captures both start (-s) and end (-t) timestamps:

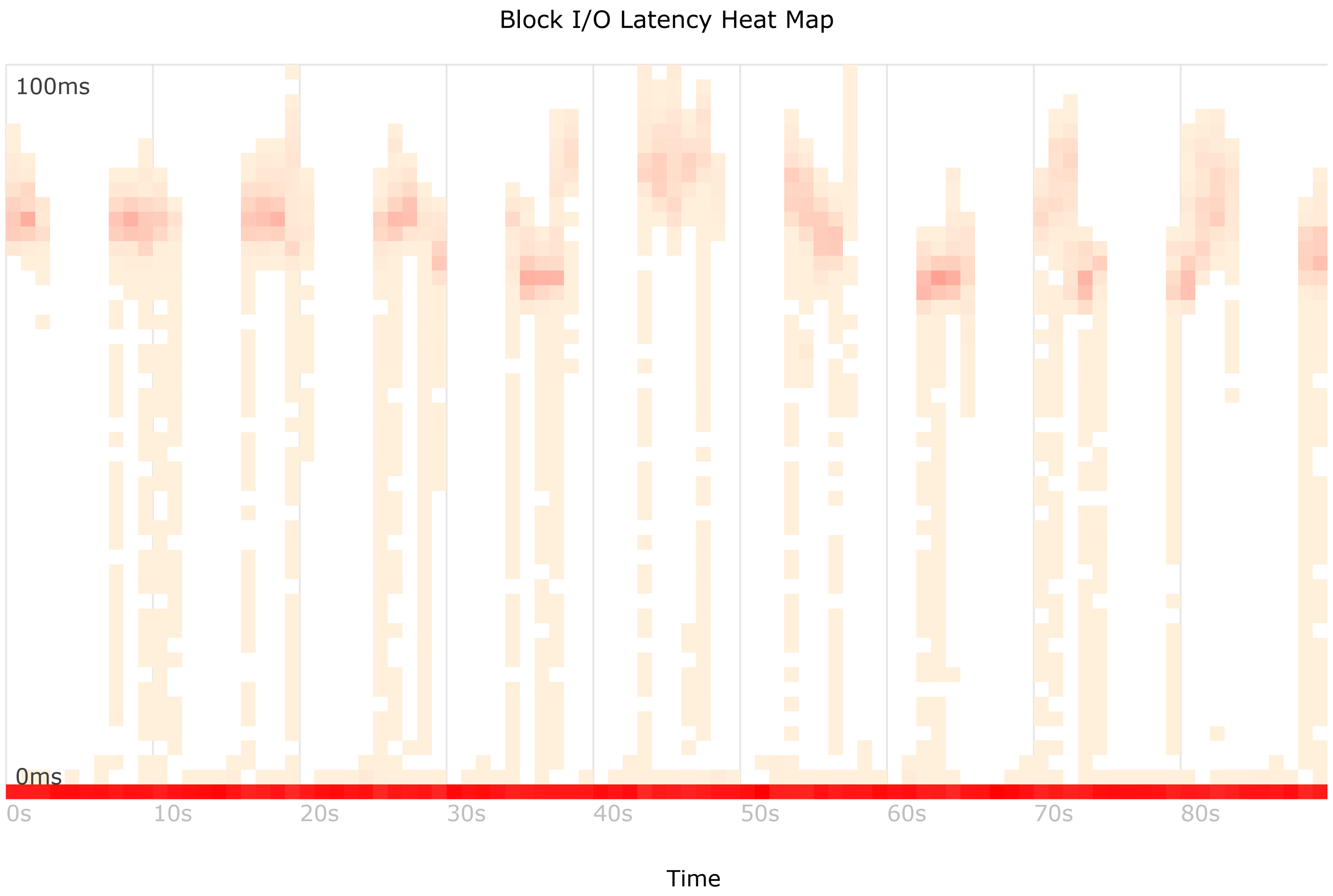
```
# ./iosnoop -ts 90 > out.iosnoop
# more out.iosnoop
Tracing block I/O for 90 seconds (buffered)...
STARTs         ENDs           COMM         PID    TYPE DEV      BLOCK        BYTES     LATms
6743904.592147 6743904.592316 java         9823   R    202,32   17266904     8192       0.17
6743904.594729 6743904.594907 java         9823   R    202,16   23030152     8192       0.18
6743904.597172 6743904.597402 java         9823   R    202,32   1405848      8192       0.23
6743904.598571 6743904.598745 java         9823   R    202,32   25259784     8192       0.17
[...]
```

I won't use the start timestamp (STARTs), but having it at high resolution may be useful for later study.

Now converting this into a heatmap:

```
# git clone https://github.com/brendangregg/HeatMap
# cd HeatMap
# cat ../out.iosnoop | awk '$1 ~ /^[0-9]/ { print $2, $9 }' | ./trace2heatmap.pl \
    --unitslatency=ms --unitstime=s --maxcol=90 --maxlat=100 --grid \
    --title="Block I/O Latency Heat Map" > biolatencyheatmap.svg
```

You can tune the various options as desired. I truncated it to 100 milliseconds (--maxlat), and added a title. This makes the following (mouse-over for details, or try the direct SVG):



Great! I use these to examine latency outliers, as well as the distribution for the bulk of the I/O. Time is on the x-axis, and I/O latency on the y-axis. The number of I/O at each time and latency range is shown by the darkness of each block: darker for more.

In this case the bulk of the I/O is very fast, between 0 and 2 milliseconds (shown as the dark red line at the bottom). There are also clouds of high latency I/O, about every 9 seconds, which are around 70 milliseconds, creating a multimodal distrubition. Their presence would be difficult to see from average latency alone.

It turns out that these are due to a single disk in particular, which I can filter using awk:

```
# cat ../out.iosnoop | awk '$6 == "202,1" { print $2, $9 }' | ...
```

I also reduced the queue size to this slow disk using:

```
# echo 4 > /sys/block/xvda1/queue/nr_requests
```

This type of tuning was suggested in my previous blog's comments, as a possible relief for I/O latency outliers caused by reads queueing behind a large batch of writes. This reduced the severity of the I/O latency outliers a little, and the I/O clouds a lot. (This tuning will also hurt performance for that one disk, so don't copy it without understanding what it does.)

See the before (128 queue length) and after (4 queue length) latency heat maps for that disk.

Have disk I/O issues? Aren't using latency heat maps? You should! See my heat maps page for more details.

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).