acmqueue

Visualizing System Latency

Heat maps are a unique and powerful way to visualize latency data. Explaining the results, however, is an ongoing challenge.

Brendan Gregg, Oracle

When I/O latency is presented as a visual heat map, some intriguing and beautiful patterns can emerge. These patterns provide insight into how a system is actually performing and what kinds of latency end-user applications experience. Many characteristics seen in these patterns are still not understood, but so far their analysis is revealing systemic behaviors that were previously unknown.

INTRODUCTION TO LATENCY

Latency is time spent waiting. It has a direct impact on performance when induced by a synchronous component of an application request. This makes interpretation straightforward—the higher the latency, the worse the performance. Such a simple interpretation is not possible for many other statistics types that are commonly examined for performance analysis, such as utilization, IOPS (I/O per second), and throughput. Those statistics are often better suited for capacity planning and for understanding the nature of workloads. For identifying performance issues, however, understanding latency is essential.

For application protocols measured from the application server, latency can refer to the time from when a request was received to when the completion was sent—for example, the time for a Web server to respond to HTTP GETs or a file server to respond to NFS (network file system) operations. Such a measurement is extremely important for performance analysis since the client and end users are usually waiting during this time.

For resource components such as disks, latency can refer to the time interval between sending the I/O request and receiving the completion interrupt. High disk latency often translates to application performance issues, but not always: file systems may periodically flush dirty cached data to disks; however, the I/O is asynchronous to the application. For example, the Oracle Solaris ZFS file system periodically flushes transaction groups to disks, causing a spike in average disk latency. This does not reflect the file-system latency experienced by ZFS consumers, since the average disk latency includes asynchronous writes from the transaction flush. (This misconception would be alleviated somewhat if read and write latency were observed separately, since the transaction flush affects write latency only.)

While it's desirable to examine latency, it has been historically difficult or impossible to measure directly for some components. For example, examining application-level latency server side may have involved instrumenting the application or examining network packet captures and associating request to response. With the introduction of DTrace,[1] however, measuring latency at arbitrary points has become possible for production systems—and in realtime.

LATENCY HEAT MAPS

Given the ability to trace latency at arbitrary points of interest, the problem becomes effective visual presentation of this data. Busy systems can be processing hundreds of thousands of I/O events per second, each one providing a completion time and I/O latency. One approach is to summarize the data as average and maximum latencies per second, which can be presented as line graphs. While this would allow average latency to be examined over time, the actual makeup or distribution of that latency cannot be identified beyond a maximum, if provided.

To examine a distribution over time, visualizations such as heat maps may be used. The use of heat maps in system observability tools has been infrequent, with some appearances to map the access pattern of disk I/O. An example of this is taztool (1995), which displays a heat map showing time on the x-axis and disk I/O offset on the y-axis, allowing random and sequential disk I/O patterns to be identified by visualizing the location of disk I/O.[5]

To visualize the distribution of latency over time, a heat map can be created with time on the x-axis and latency on the y-axis. The heat map is a color-shaded matrix of pixels, where each pixel represents a particular time and latency range. The amount of I/O that occurs in that time and latency range is shown by the color shade of the pixel: darker colors for more I/O, lighter colors for less. Apart from showing the latency distribution, the heat map also conveys details on maximum and average latency by looking for the pixel with the highest latency and where the darkest colors are grouped.

For the latency heat map to be most effective, the time and latency ranges represented by each pixel should be sufficiently large to allow multiple I/O operations to fall within them. This allows darker shades to be selected and patterns shown by different shades to be observed. If the ranges are too small, many of the pixels may represent only one I/O, and much of the heat map may appear in the same color shade; it may also reduce the likelihood that adjacent pixels are shaded, and the heat map may look more like a scatter plot.

The range of possible color shades from light to dark may be applied to each heat map generated. This can be applied linearly: the pixel with the most I/O is assigned the darkest color, and all other pixels are given a shade that is scaled from the darkest I/O count. A drawback with this approach is that important details may appear washed out. Latency deviating from the norm is particularly important to examine, especially occurrences of high latency. Since these may represent only a small fraction of the workload—perhaps less than 1 percent—the color shade may be very light and difficult to see. A false color palette can be applied instead to highlight these subtle details, given the tradeoff that the color shades then cannot be used to gauge relative I/O counts between pixels.

A particular advantage of heat-map visualization is the ability to see outliers. For the latency heat map these may be occasional I/O operations with particularly high latency, which can cause significant performance issues. If the y-axis scale is automatically picked to display all data, outliers are easily identified as occasional pixels at the top of the heat map. This also presents a problem: a single I/O with high latency will rescale the y-axis, compressing the bulk of the data. When desired, outliers can be eliminated so that the bulk of the I/O can be examined in detail. An automatic approach can be to drop a percentage (say, 0.1 percent) of the highest-latency I/O from the display, when desired.

To generate latency heat maps, data is collected for each I/O event: the completion time and I/O latency. This data is then grouped into the time/latency pixels for the heat map, and the pixels

are shaded based on their I/O counts. If the original I/O event data is preserved, heat maps can be regenerated for any time and latency range, and of different resolutions. A problem with this is the size of the data: busy production systems may be serving hundreds of thousands of I/O events per second. Collecting this continually for long intervals, such as days or weeks, may become prohibitive—both for the storage required and the time to process and generate the heat maps. One solution is to summarize this data to a sufficiently high time and latency resolution and to save the summarized data instead. When displaying heat maps, these summaries are resampled to the resolution desired.

HEAT MAP EXPLAINED

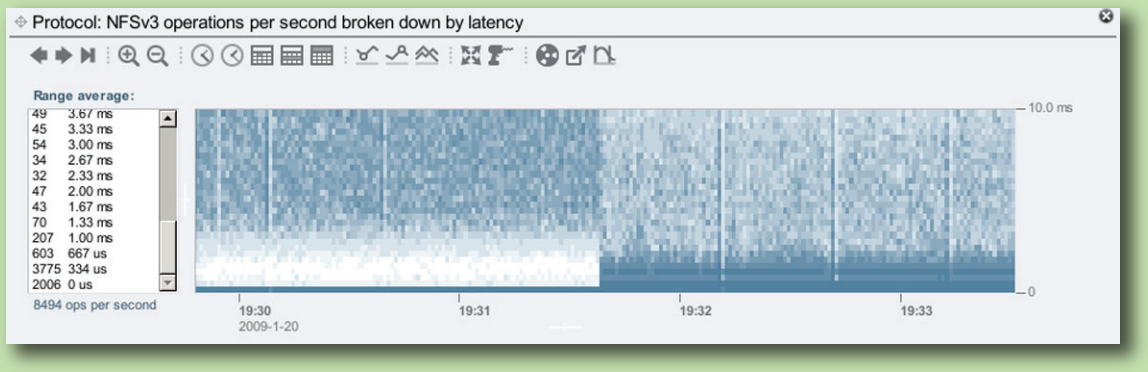
Latency heat maps were implemented as part of a system-observability tool called Analytics. The implementation allows them to be viewed in realtime and continually records data with a one-second granularity for later viewing. This is made possible and optimal by DTrace, which has the ability to trace and summarize data in-kernel to a sufficient resolution and to return these summaries every second to user-land. The user-land software then resamples the summarized data to produce the heat maps.

The heat map in figure 1, an example screenshot from Analytics, shows the latency distribution of an NFS read workload and the effect on NFS latency when using an additional layer of flash-memory-based cache. This cache layer was enabled at 19:31:38, which has been centered on the x-axis in this screenshot. Explaining this heat map in detail will show how effective this visualization is for understanding the role of these system components and their effect on delivered NFS latency.

In this screenshot, a panel is displayed to the left of the heat map to show average IOPS counts. Above and below the panel the “Range average:” and “8494 ops per second” show the average NFS I/O per second for the visible time range (x-axis). Within the panel are averages for latency ranges, the first showing an average of 2,006 NFS IOPS between 0 and 333 µs. Each of these latency ranges corresponds to a row of pixels on the heat map.

For the time before 19:31:38, the system served NFS reads from one of two locations: a DRAM-

FIGURE 1 **NFS Latency When Enabling SSD-based Cache Devices**



based cache or disk storage. If the requested I/O was not in the DRAM cache, then it was retrieved from disk instead. In the heat map, two levels of latency can be seen. These correspond to:

- DRAM hits, shown as a dark line at the bottom of the heat map
- Disk hits, shown as a shaded cloud of latency from 2 ms and higher

This is as expected. DRAM hits have very low latency and are shown in the lowest-latency pixel. This pixel represents latencies between 0 and 333 µs, which is the resolution limit of the currently displayed heat map. Since the recorded data has a higher resolution, this heat map can be redrawn with different vertical scales to reveal finer details. By zooming to the lower latencies the DRAM hits were found to be mostly in the range of 0 to 21 µs.[2]

The latency for disk hits has a wide distribution, from about 2 ms to the top of the displayed heat map at 10 ms. The returned latency for disk I/O includes rotation, seek, and bus I/O transfer times. As the disks were accessed with a random I/O pattern, rotational latency alone can add up to 8.3 ms, the time for a full rotation on these disks. This rotational latency is presumed responsible for much of the random pattern seen in the heat map.

The heat map before 19:31:38 also identifies a latency range where I/O is less frequent: the lighter band seen from 334 µs to about 2 ms, between DRAM hits and disk hits. This latency gap has been addressed by the hybrid storage pool[4] in ZFS[6] by adding a flash-memory-based layer of cache. Flash memory is slower than DRAM and faster than disks, and was incorporated in this NFS server in the form of SSDs (solid state disks). NFS reads may then be served from one of three locations, in order of preference: the DRAM-based cache, the flash-memory-based cache, or disk storage.

Enabling the flash-memory-based cache occurred at 19:31:38, after which three levels of latency can be seen:

- DRAM hits, shown as a dark line at the bottom of the heat map
- SSD hits, having a latency of less than about 2 ms
- Disk hits, which have become lighter with the addition of the extra cache layer, since fewer requests are reaching disk

This heat map shows that a flash-memory-based cache had reduced latency for I/O that would otherwise be served from disk. All three system components were visualized, with their latency ranges and the distribution of latency within that range. It also shows that disk I/O still occurs, although at a reduced rate. This is all useful information provided by the heat-map visualization. Imagine presenting this data as a line graph of average latency instead: the only information visible would be a small reduction in average latency when the cache was enabled (small since the average would be dominated by the high number of DRAM hits).

Most heat maps are well understood like this one. What follows are heat maps we have discovered that were not expected and that exhibit interesting patterns that are not fully understood.

THE ICY LAKE

The workload and target are simple: A single client has a single thread performing sequential synchronous 8-KB writes to an NFS share. The NFS server has 22 x 7,200-RPM disks as part of a ZFS striped pool.

Since these are synchronous writes, the NFS request cannot complete until the data is written onto stable storage. No flash-memory-based log devices were used for this test, so latency is expected to be high, as the data must be written to the 7,200-RPM disks.

You may expect the latency to be distributed randomly between 0 and 10 ms and for the heat map to appear as white noise. The actual result is shown in figure 2.

Instead of a random distribution, latency is grouped together at various levels that rise and fall over time, producing lines in a pattern that became known as the icy lake. This was unexpected, especially considering the simplicity of the workload.

FIGURE 2

Synchronous Writes to a Striped Pool of Disks

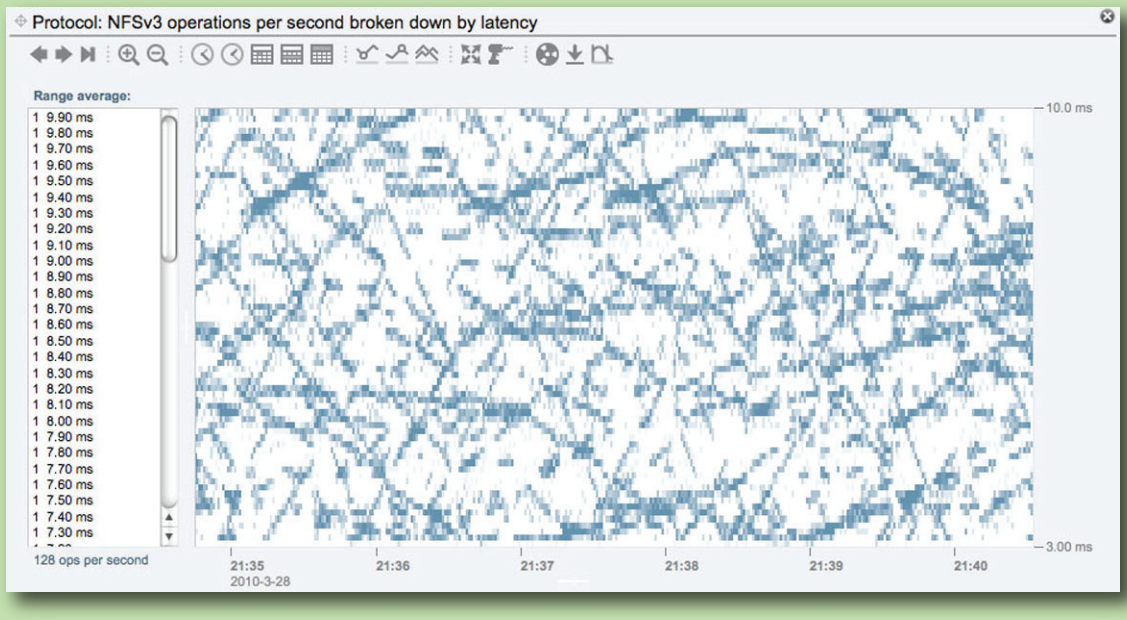
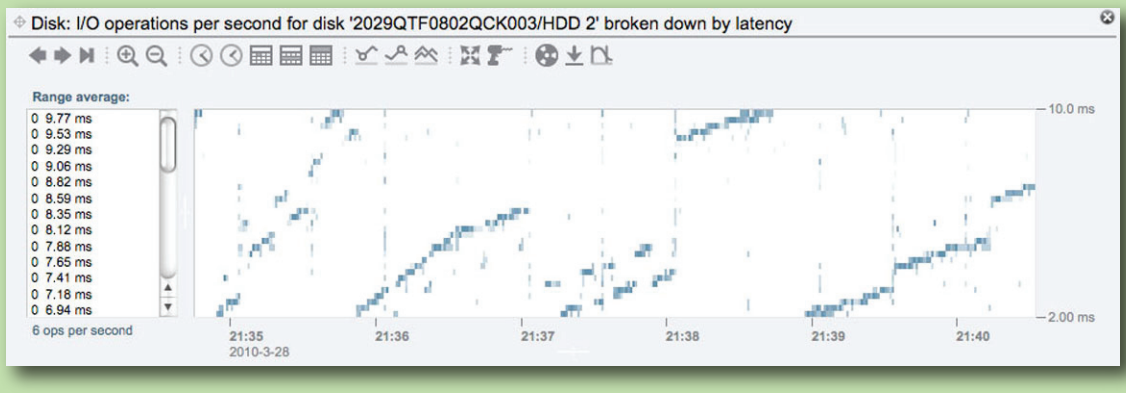


FIGURE 3

Single Disk Latency from a Striped Pool of Disks



This behavior could not be identified from average or maximum latency alone—imagine compressing the y-axis information into a single line graph. This would also be challenging to identify when examining every I/O event, such as by tracing at the disk level using the DTrace-based iosnoop tool, because of the sheer volume of the data (thousands of lines of output).

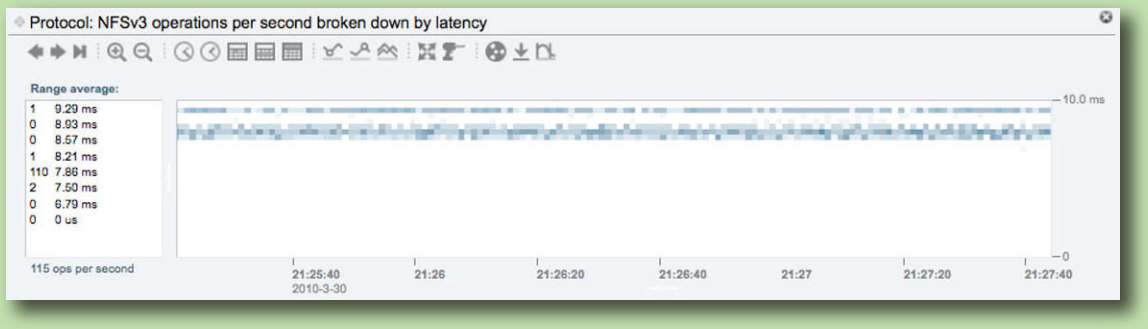
The first step in understanding this pattern is to check if each of the 22 disks contributed distinct lines. Figure 3 shows the disk I/O latency from a single disk, which confirms that each disk is contributing lines to the pattern.

The next step is to investigate why some lines increase and some decrease. An increase could result from an application requesting I/O in lockstep with disk rotation and ZFS writing sectors along tracks on disk, increasing disk-rotation latency with each I/O (although this doesn't explain how latency could increase for some disks and decrease for others).

To simplify matters, the test was repeated with a single disk pool. Figure 4 shows that most of the NFS latency was between 7.86 and 8.20 ms, which is close to the 8.33-ms rotation speed of the 7,200-RPM disk. The disk I/O offsets were examined (using both Analytics and iosnoop), which showed that ZFS was writing the 8-KB I/O sequentially across the disk. The reason for the smaller NFS latency may be the client and network latency: once one I/O completes, the disk continues to turn while the NFS completion is sent to the client; the client processes it, requests the next write, and then the next write is requested to the disk. By the time this has happened, the disk has rotated a little and so doesn't require a full rotation to write out the next offset. This would explain most of the I/O shown in the heat map; however, the reason for the line at the top is still unknown (it shows an average of one I/O per second from 9.29 ms to 9.64 ms and is made clearly visible by the false color palette used by Analytics).

ZFS serves synchronous writes by writing to ZILs (ZFS intent logs), which are later grouped and flushed to disk as a TXG (transaction group). The ZIL is expected to be written sequentially, and so the heat map is also as expected (with the exception of the line at the top). This will differ for a two-disk striped pool, since ZFS will have a ZIL on each disk and write to them in round-robin fashion. This was tested, and figure 5 shows the resultant NFS latency on a two-disk pool and the disk I/O latency from each disk in the pool. The reason for increasing and decreasing latency can now be

FIGURE 4 **Synchronous Write Latency to a Single-disk Pool**



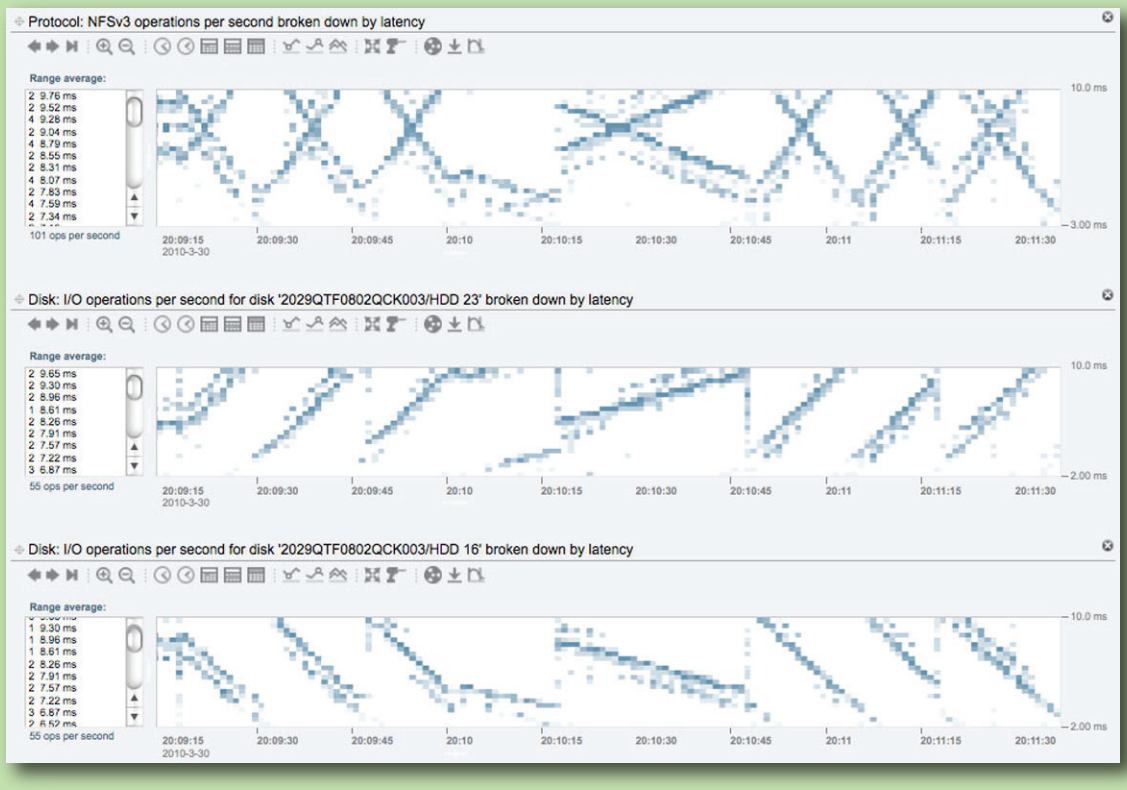
theorized: as the latency on one disk increases, the other disk continues to turn and by the time a request is issued has a corresponding smaller latency. The heat map in figure 2 is an extension of this, with 22 disks instead of two.

The reason for the slope in the first place has still not been pinpointed. The disks are writing to a steadily increasing offset, which is expected to be placed in a sequential manner along a disk track (it's up to the disk what it really does with it). If the starting point of rotation were fixed, the rotational latency to each write would steadily increase as the disk turns farther to reach the increasing offset (until a full revolution is reached). The starting point isn't fixed, however; rather it is the end point of the previous I/O, which includes the starting offset and I/O size. Since each offset delta and I/O size is the same, the rotational latency to the next I/O should also be the same. As with the single-disk pool analysis, the slope may actually be a result of client and network latency while the disks continue to rotate. The reason the slope changes is also unknown, as seen in figure 5 between 20:10:00 and 20:10:45.

This workload was tested on other storage configurations such as mirroring, single-, and double-parity RAID. Figure 6 shows this workload to a mirrored pool of 22 disks. Here the ZIL is mirrored across pairs of disks, and writes to stable storage are not considered completed until the ZIL exists on both sides of the mirror; thus, the NFS I/O latency is from the slowest disk in the pair. This has given

FIGURE 5

Synchronous Write Latency to a Two-disk Pool



the heat map a bias toward the higher latencies. A similar and greater effect was seen for single- and double-parity RAID (their heat-map screenshots are not included here).

To summarize what we know about the icy lake: lines come from single disks, and disk pairs cause increasing and decreasing latency. The actual reason for the latency difference over time that seeds this pattern has not been pinpointed; what causes the rate of increase/decrease to change (change in slope seen in figure 5) is also unknown; and, the higher latency line seen in the single-disk pool (figure 4) is also not yet understood. Visualizing latency in this way clearly poses more questions than it provides answers.

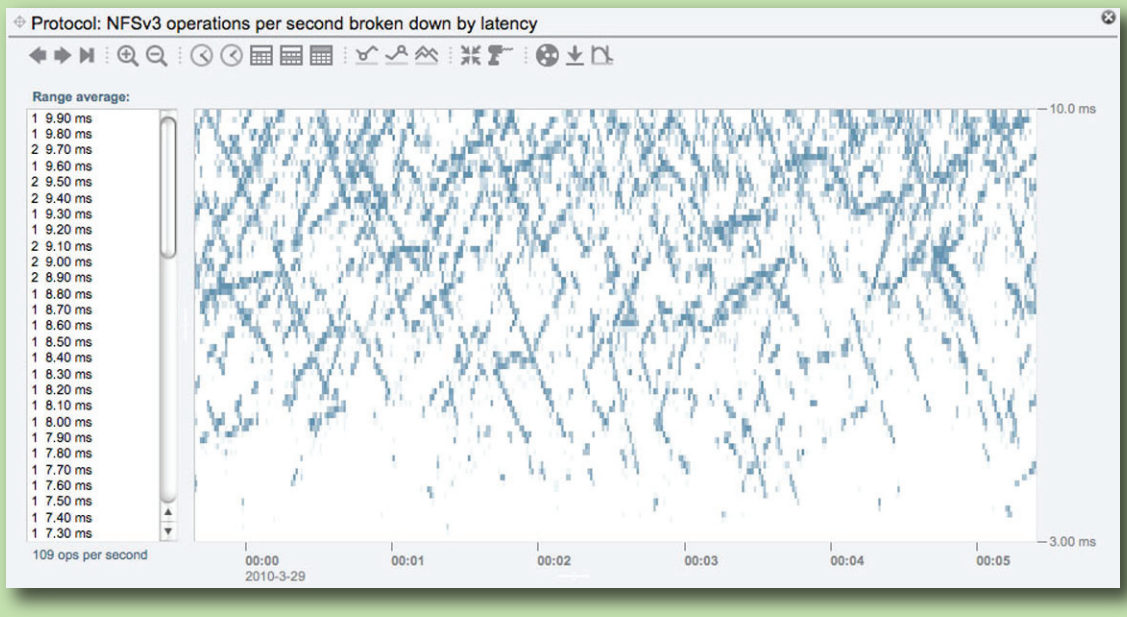
THE RAINBOW PTERODACTYL

As with the icy lake, the rainbow pterodactyl is another simple workload that has produced a surprisingly complex pattern. This time disk I/O latency is examined on a system with 48 disks across two JBOD (just a bunch of disks) enclosures. A local workload was executed to investigate I/O bus throughput by adding disks one by one with sequential 128-KB reads, while looking for knee points in the throughput graph. The latency was expected to be consistent for the I/O size used and appear as a narrow line, perhaps with a slight increase as contention increased on I/O subsystem buses (which include HyperTransport, PCIe, and SAS). When one of those buses reaches saturation, the latency is expected to increase much more sharply. Therefore, only two features were expected: a gradual increase with consistent latency and later a sharp increase with latency becoming less consistent because of contention.

Figure 7 shows the throughput graph and latency heat map from this test. A new disk was added

FIGURE 6

Synchronous Writes to a Mirrored Pool of Disks



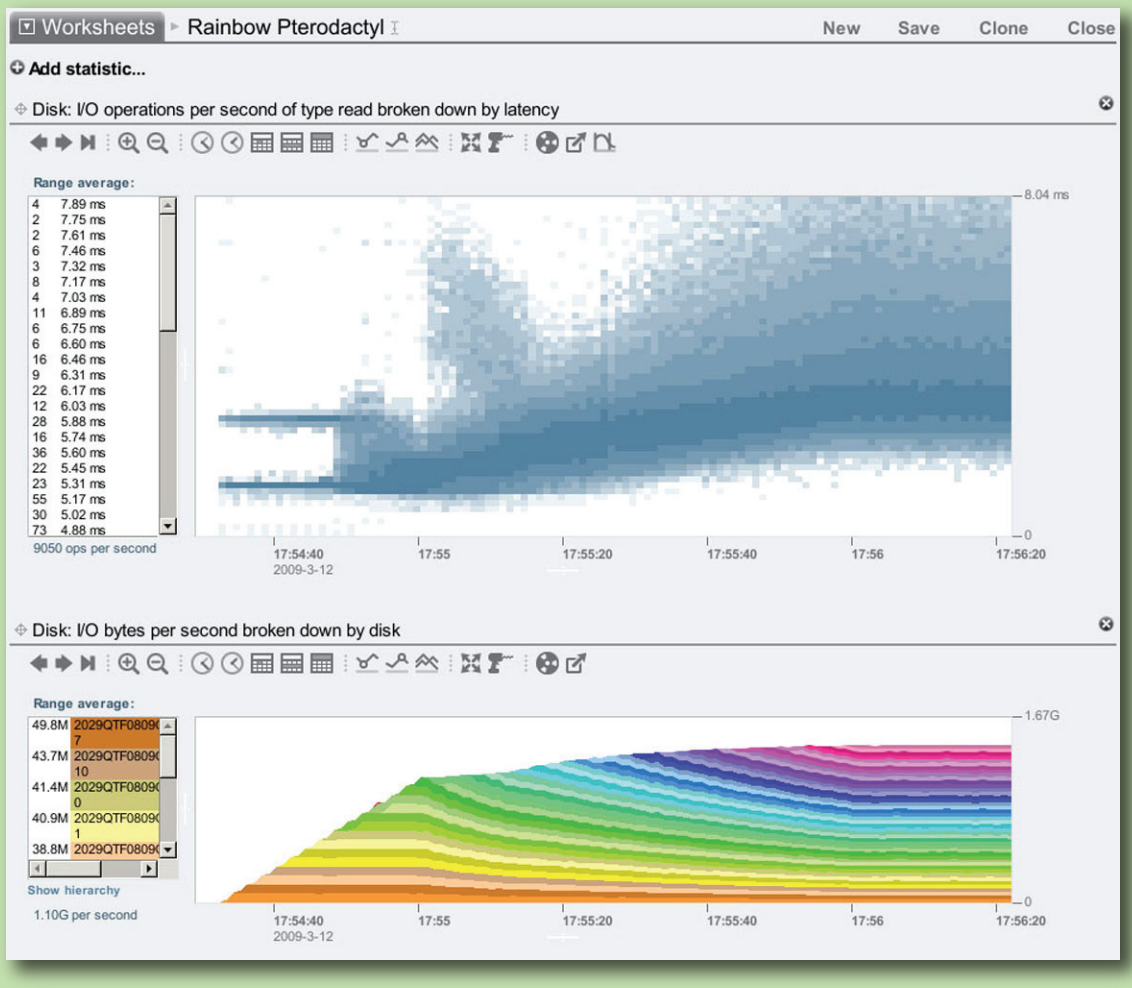
to the workload every two seconds, and a knee point in the disk throughput plot can be seen at 17:55. Finding this knee point was the original intent of this experiment; it was the latency heat map that was eye-catching, however. We named it the rainbow pterodactyl.

The disk I/O bytes graph (the rainbow) shows three features: the initial rise, followed by a decreased slope, and then decay. By corresponding the disk graph with the heat map, characteristics in the heat map can be seen to occur at certain disk counts. The heat map shows the following features.

The "beak" occurs from disk one to disk eight. The reason for two levels of latency is not fully understood, but an experiment has provided a clue: if the same data is read repeatedly to ensure disk-cache hits, then only one line is seen with low latency. The two-line pattern happens when these disks are read sequentially, suggesting that the second line is for disk-cache misses. Analyzing this further is difficult with standard tools: input to the disk and its returned latency can be

FIGURE 7

Sequential Disk Reads, Stepping Disk Count



traced, but there is no visibility into disk internals such as the operation of the disk data controller.

When the ninth disk is added, the beak turns into the "head." The disks are attached using two SAS cables, each x4 ports, providing eight SAS ports in total. Accessing the ninth disk may be causing contention on those ports in the SAS controller and the corresponding random latency pattern. When the disks are attached using a single x4 SAS cable, the beak-to-head transition occurs at the fifth disk.

A "bulge" forms at the top of the head between disks 9 and 12, showing slightly increased latency. The reason for this is not certain, though it may be from increasing contention for the SAS ports. Nor is the reason known for the reduced latency that forms the "neck" at disks 13 and 14.

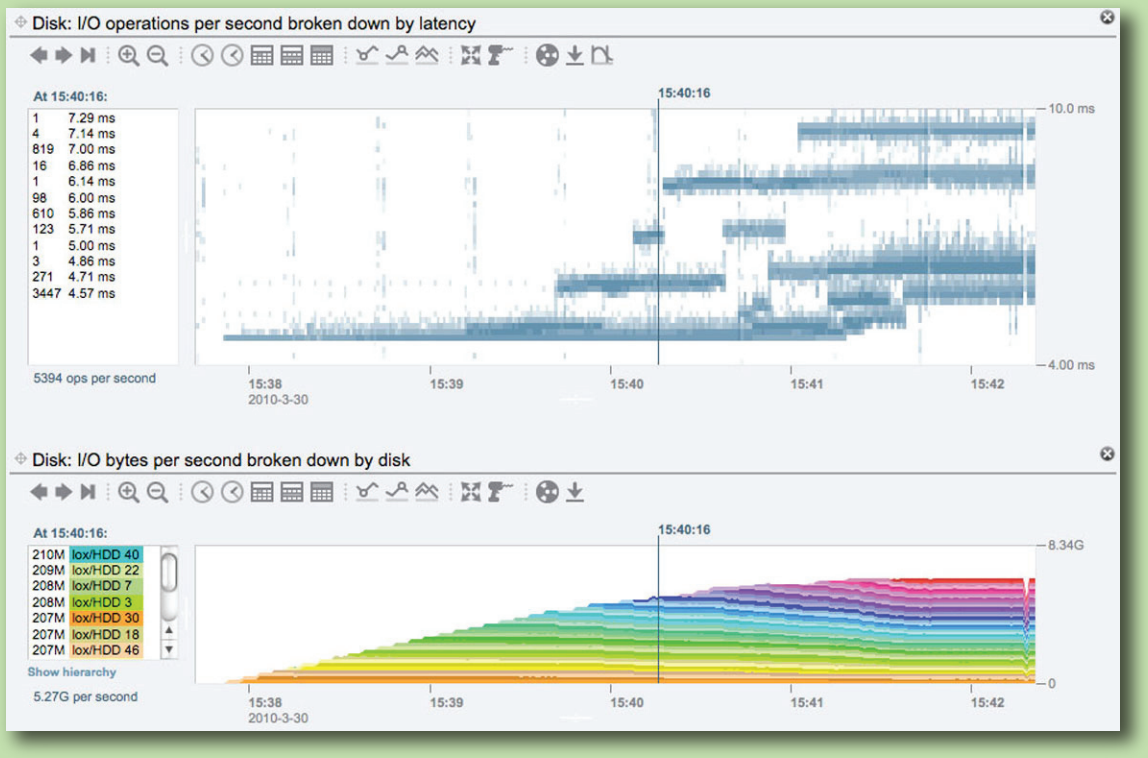
Roughly between disks 15 and 20 is the "wing." This sudden increase in latency causes the knee point in the disk-throughput graph. The source for this contention is not known, although another disk-scaling experiment using a single x4 SAS cable to a single JBOD produced a wingless pterodactyl.

From about disk 20 onward, while disks continue to be added, latency continues to rise and becomes less consistent. This is expected to be PCIe-gen1 bus contention on the SAS controller card.

All of these features are made visible by the heat map, yet are completely unknown by the individual I/O events that form the input: they provide only completion times and I/O latency, while the disk count is increased. The heat map has imaged the I/O subsystem from this data, showing components that are suspected to be disk caches, SAS ports, and the PCIe bus.

FIGURE 8

Repeated Disk Reads, Stepping Disk Count



To summarize the rainbow pterodactyl: little is known with accuracy, and much more investigation is needed. What this does show is how deep a simple visualization can become.

LATENCY LEVELS

For the rainbow pterodactyl, I/O bus throughput was tested by stepping a sequential disk-read workload. This was repeated on a different system with a more powerful I/O subsystem, and it was found that sequential disk reads from all available disks could not reach I/O bus saturation (no knee point). To see if a limit could be found, the workload was changed to read the same 128 KB from each disk repeatedly, so that each could provide more throughput only by returning from its cache. The result is shown in figure 8.

A knee point was reached between 15:39 and 15:40, although it is difficult to see in the disk bytes graph. At this point, a level of increased latency appears; a little later, there is another level (which was selected in this screenshot). At various points it appears as though a latency level has been promoted to a higher level. This was recently discovered and so far is not clearly understood. It's provided here as another example of unexpected details that latency heat maps have unearthed.

SHOUTING AT JBODS

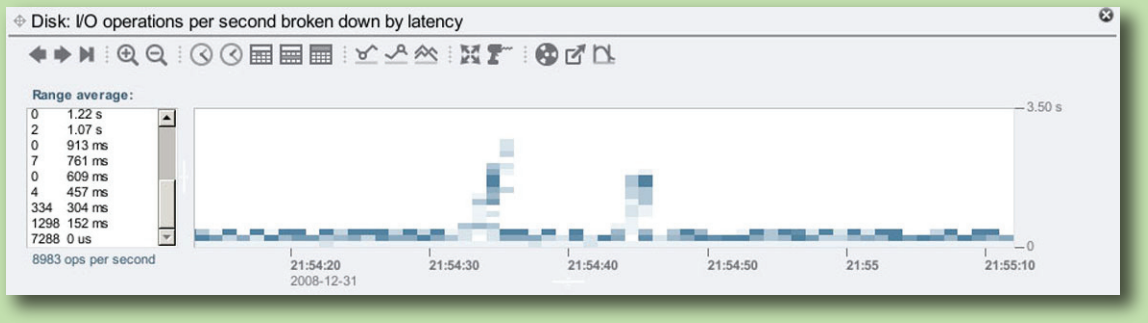
Although not as beautiful as the previous examples, the story behind the next heat map has gained some notoriety and is worth including to stress that this was a latency heat map that identified the issue.

The system included several JBODS with dozens of disks and was performing a streaming write workload. I discovered that if I shouted into the JBODs as loud as I could, the disks returned I/O with extremely high latency. Figure 9 shows the heat map from this unusual test.

The heat map shows two spikes in latency, corresponding to each of my shouts. We videotaped this discovery and uploaded it to YouTube, where I describe the effect as disk vibration.[3] It has since been suggested that this is better described as shock effects, not vibration, because of the volume of the shouts.

The affected disk I/O shown in the heat map has very high latency—more than one second. If average latency were tracked instead, a few high-latency I/O events may be drowned out on a system

FIGURE 9 **High-latency I/O**



performing more than 8,000 faster I/O events at the same time. The lesson from this experience was how well latency heat maps could identify this perturbation.

OTHER APPLICATIONS

The previous examples showed latency heat maps for systems deploying the ZFS file system, accessed over NFS. Latency heat maps are also applicable for other local and remote file system types (e.g., UFS, HFS+, CIFS), where characteristics can be identified and interpreted in similar ways. For example, UFS (Unix file system) as deployed on Solaris executes a thread named fsflush to periodically write dirty data to disk. This can update the UFS cylinder group blocks that are spaced across the disk, resulting in high-latency I/O resulting from seek and rotational latency. On older versions of Solaris the interval between writing was five seconds (tune_t_fsflushr), so on a latency heat map of disk I/O this may be easy to identify, appearing as bursts of high latency spaced five seconds apart.

The heat-map visualization can also be applied to other metrics, apart from latency. I/O size can be visualized as a heat map with size (bytes) on the y-axis, allowing any particularly large or small I/O to be identified, either of which is interesting for different reasons. I/O location can be visualized as a heat map (as mentioned earlier) with offset on the y-axis, allowing random or sequential I/O to be identified.

Utilization of components can also be visualized as a heat map showing the percent utilization of individual components, instead of displaying an average percent utilization across all components. Utilization can be shown on the y-axis, and the number of components at that utilization can be shown by the color of the heat-map pixel. This is particularly useful for examining disk and CPU utilization, to check how load is balanced across these components. A tight grouping of darker colors shows that load is balanced evenly, and a cloud of lighter pixels shows that it isn't.

Outliers are also interesting: a single CPU at 100 percent utilization may be shown as a light line at the top of the heat map and is typically the result of a software scalability issue (single thread of execution). A single disk at 100 percent utilization is also interesting and can be the result of a disk failure. This cannot be identified using averages or maximums alone: a maximum cannot differentiate between a single disk at 100 percent utilization and multiple disks at 100 percent utilization, which can happen during a normal burst of load.

All of the heat maps mentioned here have been implemented in Analytics. Along with the I/O-latency heat map, the utilization heat maps are proving to be especially useful for quickly identifying performance issues.

CONCLUSION

Presenting latency as a heat map is an effective way to identify subtle characteristics that may otherwise be missed, such as when examining per-second average or maximum latency. Though many of the characteristics shown in this article are not understood, now that their existence is known we can study them and over time identify them properly. Some of the heat maps, such as the rainbow pterodactyl, are also interesting examples of how deep and beautiful a simple visualization can be. Q

ACKNOWLEDGEMENTS

Thanks to Bryan Cantrill for developing Analytics, including the heat-map feature seen here, and co-inventing DTrace, on which this is based.

REFERENCES

1. Cantrill, B. 2006. Hidden in plain sight. *ACM Queue* 4(1): 26-36; http://queue.acm.org/detail.cfm?id=1117401.
2. Gregg, B. 2009 (Feb. 6). DRAM latency; http://blogs.sun.com/brendan/entry/dram_latency.
3. Gregg, B. 2008. Shouting in the Datacenter; http://www.youtube.com/watch?v=tDacjrSCeq4.
4. Leventhal, A. 2008. Flash storage today. *ACM Queue* 6(4): 24-30; http://queue.acm.org/detail.cfm?id=1413262.
5. taztool; http://www.solarisinternals.com/si/tools/taz/index.php.
6. ZFS; http://en.wikipedia.org/wiki/ZFS.

LOVE IT, HATE IT? LET US KNOW

feedback@queue.acm.org

BRENDAN GREGG is a principal software engineer at Oracle, where he works on performance analysis and observability in the Fishworks advanced development team. He is also the creator of the DTraceToolkit and is the co-author of *Solaris Performance and Tools* (Prentice Hall).

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.