Disk Diagnosis: Digging Deep With Monitoring Console and More

Author: David Paper, dpaper@splunk.com, @cerby on Splunk-usergroups Slack
Date: 2019-08-13

Version: 1.0.6

Table of Contents

Purpose

The purpose of this document is to offer Splunk administrators insights into the impact of storage constraints on Splunk search performance. It goes without saying that performant Splunk environments rely on sufficient resources for CPU, RAM and storage. It has been the experience of this Splunk Smokejumper that a significant number of medium to large Splunk environments are throttled by inadequate storage performance to support the demands of heavy search and indexing loads. This document enables the Splunk administrator to leverage the Splunk Monitoring Console (MC) to gather and analyze metrics external to Splunk that can indicate storage contention bottlenecks and guides the administrator on remediation options.

Problem Definition

As an I/O intensive solution, Splunk is exceptionally sensitive to storage performance issues ranging from brief transient issues (Storage Area Network (SAN) component failover, Redundant Array of Independent Disks (RAID) rebuilds) to chronic problems (not enough disk spindles, SAN capacity, port oversubscription) that can degrade Input/Output Operations Per Second (IOPS). Splunk MC provides tools to visualize disk performance at a granular level, providing the Splunk administrator with key insights into determining the optimal path to improving storage performance.

Our online documentation[1] tells us that IOPS are the primary consideration. While IOPS are an important metric, there are additional storage related metrics that paint a clearer picture of the storage subsystem. The filesystems in scope for this paper are

- $SPLUNK_HOME/var/run/splunk/dispatch
- Hot/warm bucket storage
- Cold bucket storage

These may be one, two or three filesystems.

From a Splunk administrator perspective, some of the tangible issues that can be traced to storage include:

- Search head to indexer knowledge bundle replication failures
- Event ingestion queues filling and backing up onto forwarders

[1] https://docs.splunk.com/Documentation/Splunk/latest/Capacity/Referencehardware#Reference_host_specification

- Cluster data replication issues (Search Heads or Indexers)
- Rolling restarts take a very long time to recover (return to full Replication Factor and Search Factor)

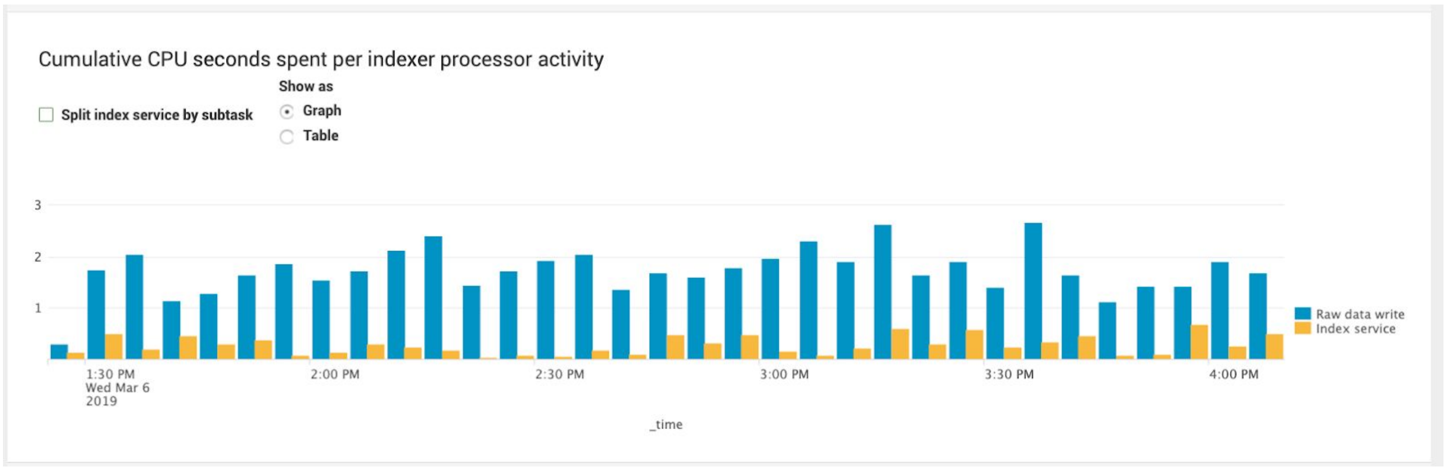
Investigation

To investigate any of the aforementioned symptoms the Splunk MC is the first place to look.

In ***MC -> Indexing -> Indexing Performance -> Indexing Performance: Instance*** toward the bottom of the dashboard is a panel titled

- Aggregate [or Cumulative, depending on Splunk version] CPU Seconds Spent per Indexer Process Activity

which breaks down the utilization metrics for raw data write and the indexing service on an indexer. If CPU raw data write usage is higher than the time Splunk spends processing events, that suggests something is wrong with storage. There can be other causes, but this is a quick metric, even if queues are not backed up, to start looking at storage and then work backwards up the chain.



For a more detailed breakdown of the storage indicators, navigate to ***MC -> Resource Usage -> Resource Usage Machine*** and select an indexer. At the bottom of the page are three panels entitled

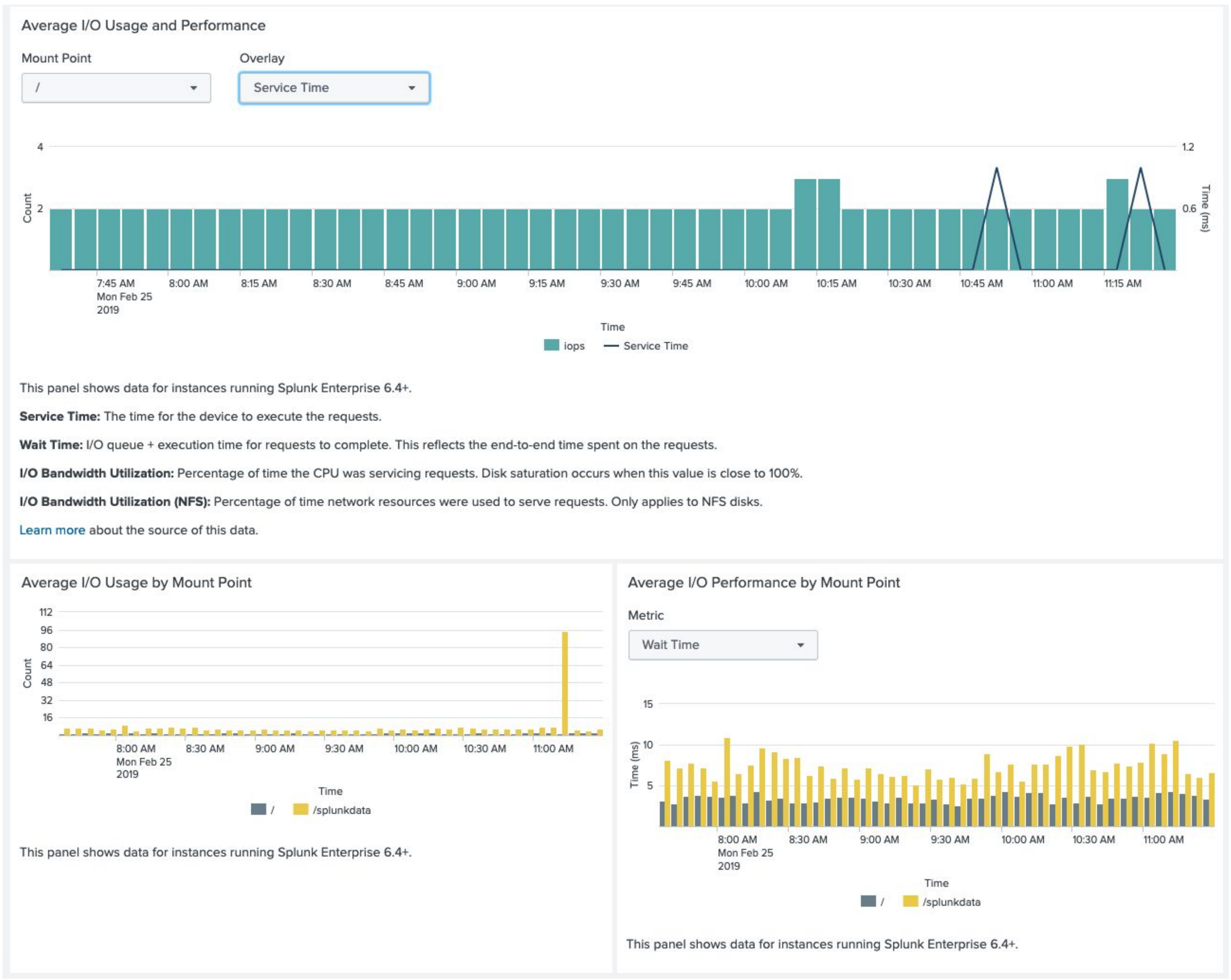
- Average I/O Usage and Performance
- Average I/O Usage by Mount Point
- Average I/O Performance by Mount Point

These MC panels are driven off _introspection data, which Splunk collects every 60 seconds and give an initial high-level indicator of issues. The IOPS, service and wait time overlays help determine if further investigation is needed.

Items to look for in the MC views above:

- IOPS performance that has unexpectedly hit a ceiling or values that should be higher based on utilization of the host (this can be hard to suss out)
- Wait times that regularly average 10 - 15ms or higher
- Utilization values that don't quickly regress back toward single-digit or low double-digit values after they spike up

In the example below, IOPS, Service Time and Wait Times are well within the good range, with Wait Times bouncing between 5 - 10ms for the /splunkdata partition.



Average I/O Usage and Performance

Mount Point: /

Overlay: Service Time

This panel shows data for instances running Splunk Enterprise 6.4+.

Service Time: The time for the device to execute the requests.

Wait Time: I/O queue + execution time for requests to complete. This reflects the end-to-end time spent on the requests.

I/O Bandwidth Utilization: Percentage of time the CPU was servicing requests. Disk saturation occurs when this value is close to 100%.

I/O Bandwidth Utilization (NFS): Percentage of time network resources were used to serve requests. Only applies to NFS disks.

Learn more about the source of this data.

Average I/O Usage by Mount Point

This panel shows data for instances running Splunk Enterprise 6.4+.

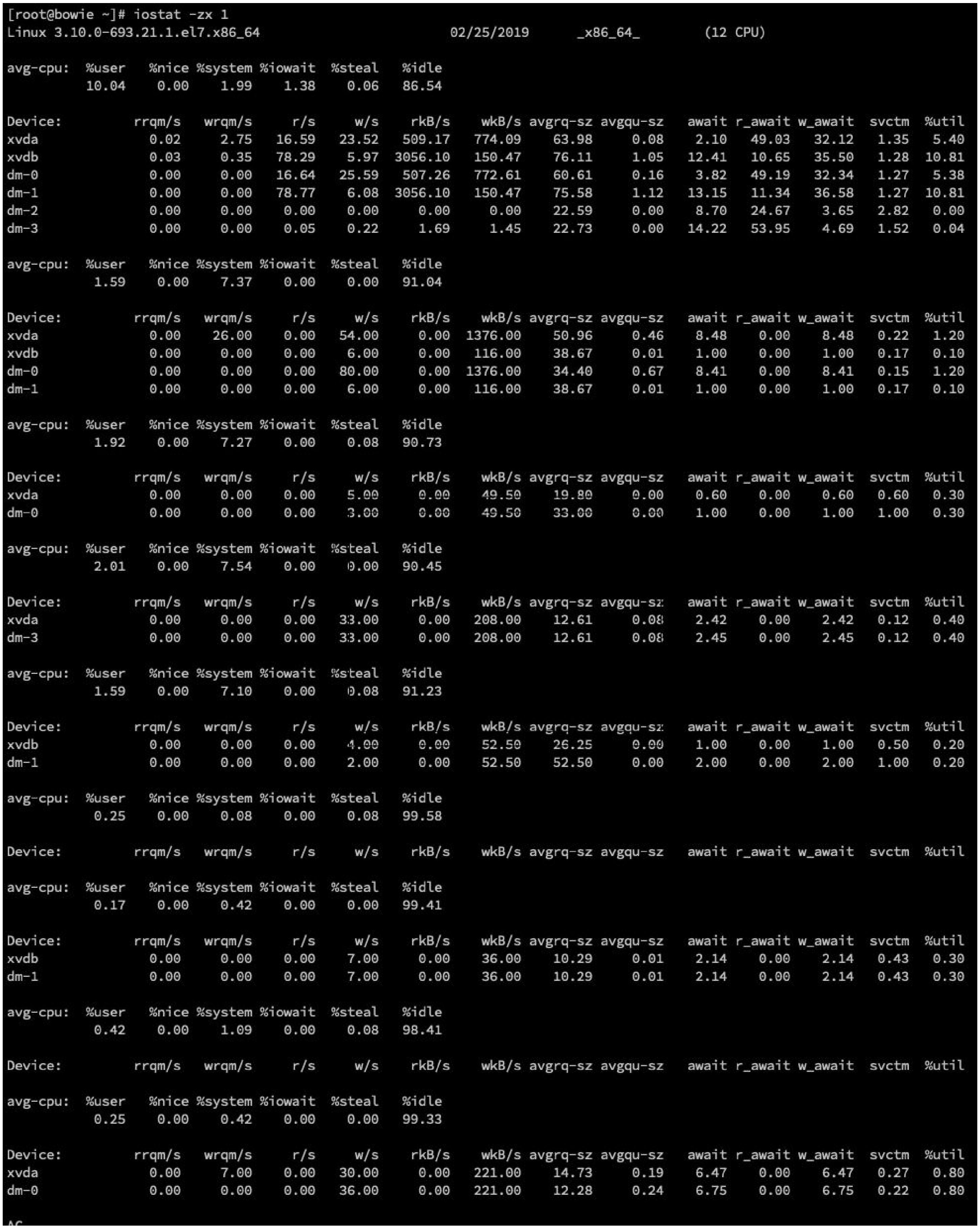
Average I/O Performance by Mount Point

Metric: Wait Time

This panel shows data for instances running Splunk Enterprise 6.4+.

Digging Deeper

On Linux servers, the sysstat (adjust for your operating system) package on RedHat Enterprise Linux based systems includes iostat, a tool to view I/O statistics at a very granular level. My favorite instantiation method is “`iostat -zx 1`” which removes any devices not active, provides extended metrics, and refreshes every second. This invocation of iostat provides a very in-depth view of I/O stats at the device and metadevice level.



```
[root@bowie ~]# iostat -zx 1
Linux 3.10.0-693.21.1.el7.x86_64                   02/25/2019      _x86_64_        (12 CPU)

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
          10.04    0.00    1.99    1.38    0.06   86.54

Device:         rrqm/s   wrqm/s     r/s     w/s    rkB/s    wkB/s avgrq-sz avgqu-sz   await r_await w_await  svctm  %util
xvda              0.02     2.75   16.59   23.52   509.17   774.09    63.98     0.08    2.10   49.03   32.12   1.35   5.40
xvdb              0.03     0.35   78.29    5.97  3056.10   150.47    76.11     1.05   12.41   10.65   35.50   1.28  10.81
dm-0              0.00     0.00   16.64   25.59   507.26   772.61    60.61     0.16    3.82   49.19   32.34   1.27   5.38
dm-1              0.00     0.00   78.77    6.08  3056.10   150.47    75.58     1.12   13.15   11.34   36.58   1.27  10.81
dm-2              0.00     0.00    0.00    0.00     0.00     0.00    22.59     0.00    8.70   24.67    3.65   2.82   0.00
dm-3              0.00     0.00    0.05    0.22     1.69     1.45    22.73     0.00   14.22   53.95    4.69   1.52   0.04

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           1.59    0.00    7.37    0.00    0.00   91.04

Device:         rrqm/s   wrqm/s     r/s     w/s    rkB/s    wkB/s avgrq-sz avgqu-sz   await r_await w_await  svctm  %util
xvda              0.00    26.00    0.00   54.00     0.00  1376.00    50.96     0.46    8.48    0.00    8.48   0.22   1.20
xvdb              0.00     0.00    0.00    6.00     0.00   116.00    38.67     0.01    1.00    0.00    1.00   0.17   0.10
dm-0              0.00     0.00    0.00   80.00     0.00  1376.00    34.40     0.67    8.41    0.00    8.41   0.15   1.20
dm-1              0.00     0.00    0.00    6.00     0.00   116.00    38.67     0.01    1.00    0.00    1.00   0.17   0.10

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           1.92    0.00    7.27    0.00    0.08   90.73

Device:         rrqm/s   wrqm/s     r/s     w/s    rkB/s    wkB/s avgrq-sz avgqu-sz   await r_await w_await  svctm  %util
xvda              0.00     0.00    0.00    5.00     0.00    49.50    19.80     0.00    0.60    0.00    0.60   0.60   0.30
dm-0              0.00     0.00    0.00    3.00     0.00    49.50    33.00     0.00    1.00    0.00    1.00   1.00   0.30

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           2.01    0.00    7.54    0.00    0.00   90.45

Device:         rrqm/s   wrqm/s     r/s     w/s    rkB/s    wkB/s avgrq-sz avgqu-sz   await r_await w_await  svctm  %util
xvda              0.00     0.00    0.00   33.00     0.00   208.00    12.61     0.08    2.42    0.00    2.42   0.12   0.40
dm-3              0.00     0.00    0.00   33.00     0.00   208.00    12.61     0.08    2.45    0.00    2.45   0.12   0.40

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           1.59    0.00    7.10    0.00    0.08   91.23

Device:         rrqm/s   wrqm/s     r/s     w/s    rkB/s    wkB/s avgrq-sz avgqu-sz   await r_await w_await  svctm  %util
xvdb              0.00     0.00    0.00    4.00     0.00    52.50    26.25     0.00    1.00    0.00    1.00   0.50   0.20
dm-1              0.00     0.00    0.00    2.00     0.00    52.50    52.50     0.00    2.00    0.00    2.00   1.00   0.20

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           0.25    0.00    0.08    0.00    0.08   99.58

Device:         rrqm/s   wrqm/s     r/s     w/s    rkB/s    wkB/s avgrq-sz avgqu-sz   await r_await w_await  svctm  %util

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           0.17    0.00    0.42    0.00    0.00   99.41

Device:         rrqm/s   wrqm/s     r/s     w/s    rkB/s    wkB/s avgrq-sz avgqu-sz   await r_await w_await  svctm  %util
xvdb              0.00     0.00    0.00    7.00     0.00    36.00    10.29     0.01    2.14    0.00    2.14   0.43   0.30
dm-1              0.00     0.00    0.00    7.00     0.00    36.00    10.29     0.01    2.14    0.00    2.14   0.43   0.30

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           0.42    0.00    1.09    0.00    0.08   98.41

Device:         rrqm/s   wrqm/s     r/s     w/s    rkB/s    wkB/s avgrq-sz avgqu-sz   await r_await w_await  svctm  %util

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           0.25    0.00    0.42    0.00    0.00   99.33

Device:         rrqm/s   wrqm/s     r/s     w/s    rkB/s    wkB/s avgrq-sz avgqu-sz   await r_await w_await  svctm  %util
xvda              0.00     7.00    0.00   30.00     0.00   221.00    14.73     0.19    6.47    0.00    6.47   0.27   0.80
dm-0              0.00     0.00    0.00   36.00     0.00   221.00    12.28     0.24    6.75    0.00    6.75   0.22   0.80

^C
```

Looking at the various columns

- Utilization % (“%util”)
    - Ideal: 0-10%
    - Good: 10-50%, with spikes higher, but returning to “normal” within 1-2 seconds
    - Bad: 50-100%, with sustained at 100% for many seconds at a time before dropping, returning to 100 within a few seconds consistently
- Disk wait times for read (“r_await”) and write (“w_await”)
    - Ideal: 0-5 ms
    - Good: 5-15 ms, with spikes higher, but returning to “normal” within 1-2 seconds
    - Bad: 15-300+ ms, higher values sustained for a short number of seconds, and then dropping down, but rarely seeing single digits
- Disk wait queues for read (“avgrq-sz”) and write (“avgqu-sz”)
    - Ideal: 0-10
    - Good: 10-20, with spikes higher, but returning to “normal” within 1-2 seconds
    - Bad: 20-300+, with higher values being present most of the time

Ideal vs. good vs. bad results are generally easy to discern. Using metrics to compare performance between indexers for ingestion rates, indexing queue fill rates, streaming bucket replication errors and search bundle replication can help determine which indexer hardware resource configurations are to be emulated and which are to be avoided. Put another way, if there are indexers in the environment that are performing well — higher than local environment average ingestion rates, low indexing queue fill rates, few or no replication errors and bundle replication issues — look beyond the Splunk configurations to figure out why storage is performing better on the good indexers compared to bad indexers.

Consider creating a new scripted input to pull granular iostats metrics into Splunk for longer term trending. The patterns that emerge may be quite enlightening.

Possible Problems

If indexers are using local disk:

- Are the disks installed those the customer thinks they have? Validate drive model numbers to determine what speed the disks are (5400 RPM vs 15k RPM vs SSD). You can use Google to determine how many IOPS each drive is capable of based on model number.
- If spinning rust, are there enough spindles?
- What RAID config is in place?

- If using hardware RAID, are there vendor advisories for performance issues with controller firmware?
- Are host, RAID controller and other firmware versions consistent across all indexers? Are they consistent between good and bad indexers?

For example, an indexer is configured with four 5400 RPM drives in a RAID 6 configuration. That configuration provides only 150 usable IOPS, and performance will be exceptionally bad.

If indexers are using shared storage or are in a virtualized environment

- Is there enough bandwidth (bytes, packets per second) available all the way to the disk?
- Are there any individual SAN ports in the path that are saturated?
- Are there any SAN ports that are logging errors, like buffer exhaustion?
- Do the disk arrays (trays) have sufficient capacity to provide IOPS to indexers mounting Logical Unit Numbers (LUN) carved from those disks?
- Are the disk array controllers having caching issues?
- Is data pinned to a non-performant tier in the SAN?
- Are there vendor advisories for performance issues with firmware or BIOS settings on storage components?

Next Steps

Depending on what is found, the problem(s) may be fixable with configuration changes (RAID group changes, LUN reprovisioning, firmware updates, BIOS setting changes), or may require hardware changes if the wrong disks are in use, there are not enough spindles, or you need RAM to take some of the load off of the I/O subsystem.

If a virtualized infrastructure and/or shared storage is in use, the storage team will need to be involved. There are limits to the amount of data able to be gleaned from the client OS layer. The storage team will need to understand the metrics being discussed and determine what is going on in the SAN environment.

If data from the storage components aren't being Splunked, it's a great time to start. Everyone loves that new use case smell.

SmartStore Considerations

SmartStore changes the read/write patterns of the hot/warm (hot) disk with the addition of the cached buckets co-existing on the same filesystem. Traditionally, writes happen at data ingestion time, including replication of hot data, until the bucket rolls to warm where it becomes read only. Based on the rate of ingestion, this write traffic is fairly steady. The rest of the disk I/O on hot is reads, which is usually the vast majority.

In SmartStore, the same ingestion based pattern exists with the addition of the cachemanager retrieving buckets from remote storage for search which significantly increases write volume. These bucket retrievals may be hundreds of megabytes per second worth of writes, as the default bucket retrieving configuration allows for eight simultaneous downloads.

Disks previously able to absorb the write load of regular ingestion and replication may not be able to handle the caching requirements of the SmartStore cachemanager. Specifically, RAID5 (one parity disk) and RAID6 (two parity disks) volumes may not be able to keep up with the throughput requirements for writes when the cachemanager is downloading multiple buckets concurrently. This I/O contention will impact search performance, ingestion and bucket replication. A RAID0 configuration for the hot disk performs better than RAID5 or RAID6. RAID10 is an acceptable choice but cuts usable disk space by 50%. Concerns about RAID0 disk failure are mitigated by maintaining the default of RF=3 which still apply to hot buckets.

Conclusion

Inadequate storage performance is one of several hardware related resources that can cause Splunk performance to suffer. If after investigating the storage environment, all metrics appear to be well within acceptable ranges, consider this a successful storage deep dive. It has succeeded in eliminating a variable from the performance troubleshooting equation.

If, on the other hand, storage is found to be the culprit, this document has helped determine there is a problem, wrapped some metrics around it, and provided enough information for the Splunk administrator to convince management that there is an opportunity to increase the ROI of the Splunk environment by addressing the problems identified. The same techniques outlined above should be used to validate the fixes along the way to a more performant Splunk environment.

Acknowledgements

Thank you to Josh Wisely, Nadine Miller, Dave Shpritz, Jimmy Maher and Erik Martinez for their feedback on this discussion.