Technical Disclosure Commons

Defensive Publications Series

July 2021

STORAGE PERFORMANCE BOTTLENECK DETECTION IN ERROR-FREE NETWORKED BLOCK STORAGE ENVIRONMENTS

Paresh Gupta

Harsha Bharadwaj

Arthur Scrimo

STORAGE PERFORMANCE BOTTLENECK DETECTION IN ERROR-FREE NETWORKED BLOCK STORAGE ENVIRONMENTS

AUTHORS:
Paresh Gupta
Harsha Bharadwaj
Arthur Scrimo

ABSTRACT

Enterprise data centers employ a distributed block storage architecture for their Tier 1 workloads. Business-critical applications are hosted on servers having local compute and memory resources, but the storage is centralized within distributed storage devices. Although a distributed block storage architecture helps an enterprise data center in efficiently using storage, when an application slowdown is experienced, the troubleshooting process becomes more difficult and manual. To address these types of challenges, techniques are presented herein that support a data-driven and algorithmic approach to pinpoint the exact root-cause (e.g., a host, a storage area network (SAN), or a storage array) of a storage slowdown. The presented techniques are operable even when no obvious errors are present (e.g., the storage access is sick-but-not-dead). The presented techniques leverage the latency metrics Exchange Completion Time (ECT), Data Access Latency (DAL), and Host Response Latency (HRL) to pinpoint the exact root-cause of the storage slowdown in a distributed block storage architecture using Small Computer System Interface (SCSI) and nonvolatile memory express (NVMe) over any transport (e.g., Fibre Channel (FC), FC over Ethernet (FCoE), Internet Small Computer Systems Interface (iSCSI), NVMe over Transmission Control Protocol (NVMe/TCP), remote direct memory access (RDMA) over Converged Ethernet (RoCE), etc.).

DETAILED DESCRIPTION

Enterprise data centers employ a distributed block storage architecture for their Tier 1 workloads. Business-critical applications are hosted on servers having local compute and memory resources, but the storage is centralized within distributed storage devices. While the servers connect to these storage devices over a storage area network (SAN), the applications remain unaware that the storage is physically outside of the server enclosure.

In virtualized environments, a hypervisor typically handles the operations for block network storage. As a result, even the virtual machine operating system is unaware that the storage is physically outside of the server enclosure. Overall, the storage access requests by the applications are transported using block storage transactions within the SAN. The application does not know that its storage requests are sent over a SAN nor does the SAN need to know the details of the applications to transport the frames between the servers and the storage device.

However, an unsolved problem remains – when a slowdown is experienced, it is often difficult to determine what caused the slowdown. Although a distributed block storage architecture may help enterprise data centers to efficiently use storage, the troubleshooting process becomes more difficult and manual when an application slowdown is experienced. The time that is taken for an application to respond, generally known as the application response time, comprises a variety of elements.

One of the major contributors to an application response is the time that is taken to access storage. At the infrastructure layer, the distributed storage has components within servers (e.g., an operating system stack, drivers, adapters, etc.), components within a network (e.g., SAN switches), and components within the storage array (e.g., a hard disk drive or a solid-state drive (SSD), controllers, adapters, etc.). To an application owner, an increase in storage response time may be the end of the troubleshooting, but at the infrastructure layer finding the root cause may still be an unanswered question.

Many factors can adversely affect an increase in the storage response time. For instance, an issue with the storage stack of the server, an issue with a storage array, or an issue within the network (for example, ongoing congestion) may be places to consider. In most enterprise data centers, pinpointing the exact root-cause of a storage slowdown is based on brute-force and trial and error methods. This is a long and manual process that can lead to excessively long outages. Existing approaches are based on the obvious symptoms of a slowdown, including for example timeout drops, errors, discards, aborts, etc.

The problem, however, often remains unsolved in the absence of these obvious symptoms. In such cases, the applications are slow but they continue to work. Similarly, storage is accessible but slow. These sick-but-not-dead symptoms are becoming unacceptable in modern all-flash and nonvolatile memory express (NVMe) storage architectures.

A block storage input/output (IO) transaction has multiple steps, and every step takes a finite amount of time. These delay values are known as IO latency metrics, of which they are three types:

- Exchange Completion Time (ECT). The amount of time that it takes to complete a read or write IO transaction.
- Data Access Latency (DAL). Latency that is introduced by a target.
- Host Response Latency (HRL). Latency that is introduced by a host.

To address the types of challenges that were described above, techniques are presented herein that leverage the ECT, DAL, and HRL metrics to pinpoint the exact root-cause of a storage slowdown. The presented techniques do not redefine the ECT, DAL, and HRL metrics. Rather, the techniques construct a new method of using the metrics, as will be described and illustrated in the narrative below. Aspects of the techniques presented herein may be comprise a series of evaluation and decision steps which may be explicated through the following description.

ECT indicates the overall storage access performance. When application response time increases and ECT does not change, it is safe to assume that storage access is not the reason for the application slowdown. On the contrary, when application response time increases and ECT also increases, it means that the storage issue is the root cause of the application slowdown. This represents a first level of pinpointing. Using this approach, the root cause of the issue is narrowed firstly to infrastructure and further to the storage layer. Read and write transactions have different ECT values. The read ECT values must be analyzed first, followed by the write ECT values.

Next, if ECT increases and DAL also increases it indicates a slowdown within the storage array. On the contrary, when ECT increases but DAL does not increase then the storage array may be ruled out. This represents a second level of pinpointing. Read and write transactions have different DAL values and they must be compared separately.

Similarly, if ECT increases and HRL also increases, this may be an indication of a slowdown within the host. However, when ECT increases but HRL does not increase, then the host may be ruled out. This represents a third level of pinpointing. This step is required for the write transactions only as an HRL metric is not available for the read transactions.

Finally, if the ECT increases for both the read and the write transactions but the respective DAL and HRL metrics do not increase, it indicates a slowdown due to SAN issues (for example, SAN congestion).

In accordance with the techniques presented herein, the correlation between ECT, DAL, and HRL must follow a unique sequence for the read and write transactions, as depicted in the flowchart that is presented in Figure 1, below.

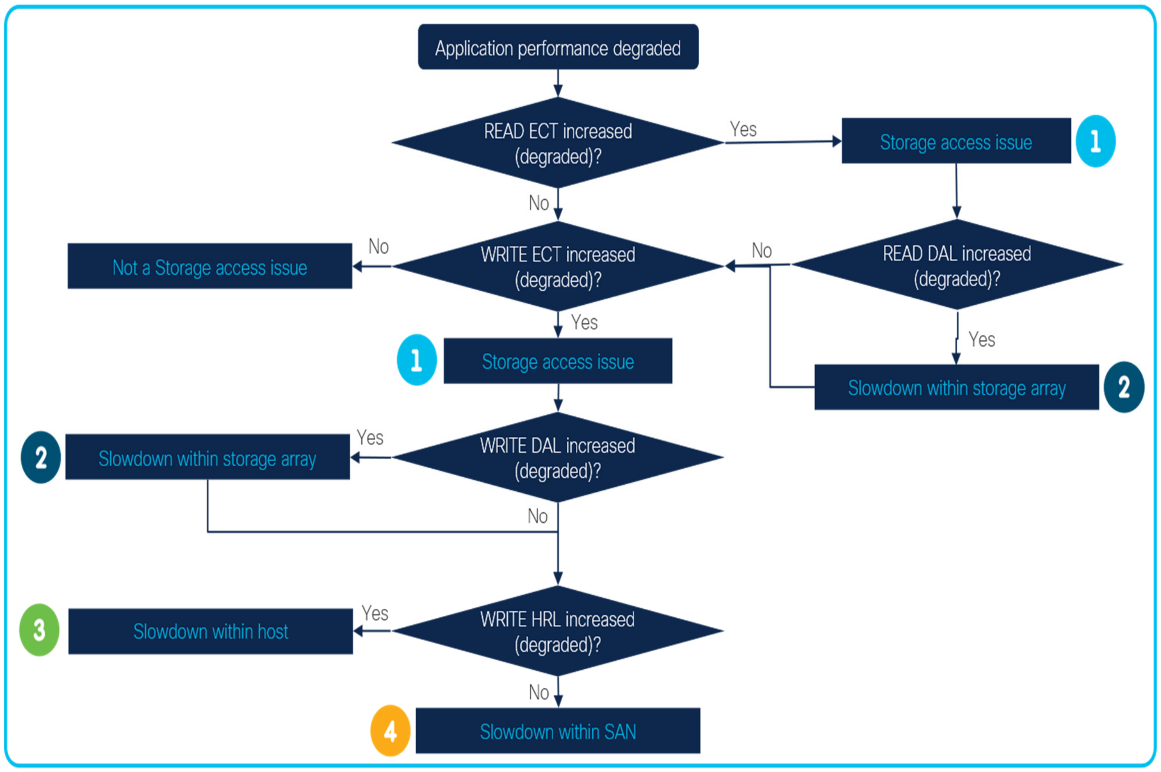
Application performance degraded
READ ECT increased (degraded)?
Yes
Storage access issue
1
No
Not a Storage access issue
No
WRITE ECT increased (degraded)?
No
READ DAL increased (degraded)?
Yes
1
Storage access issue
Yes
Slowdown within storage array
2
2
Slowdown within storage array
Yes
WRITE DAL increased (degraded)?
No
3
Slowdown within host
Yes
WRITE HRL increased (degraded)?
No
4
Slowdown within SAN

Figure 1: Illustrative Metrics Correlation Flowchart

The algorithm that as depicted in Figure 1 will, when implemented according to aspects of the techniques presented herein, result in a conceptual representation of the ECT, DAL, and HRL metrics as presented in Figure 2, below.

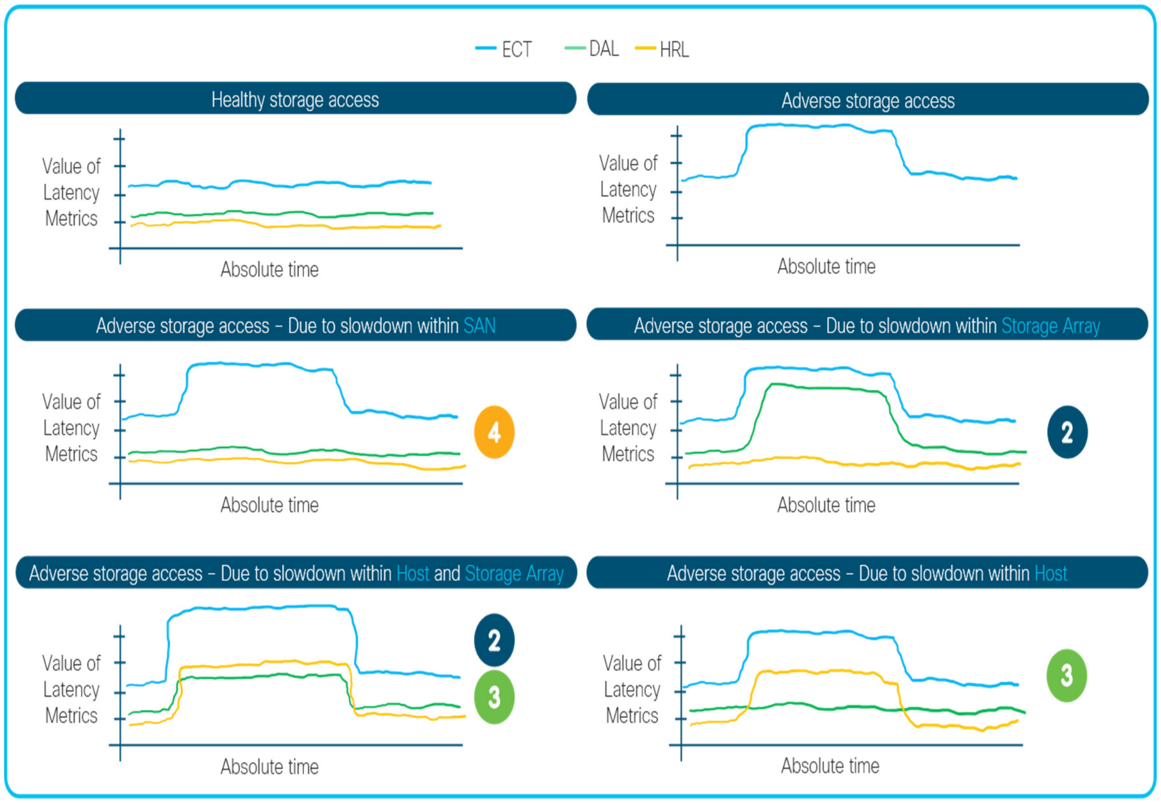

Figure 2: Illustrative Conceptual Metrics Representation

Further explication of aspects of the techniques presented herein may be made with reference to the following four implementation details. First, the same and consistent metric calculation location must be used. The IO latency metrics can be calculated anywhere in the end-to-end IO path between applications and storage. Whatever metric calculation location is selected, it must remain the same and consistent over time. For example, if ECT is calculated at storage connected ports on a SAN switch, DAL and HRL must be calculated at the exact same location. The metric calculation location must not change over time so as to avoid a change (e.g., an increase) in the values.

Second, absolute values of IO latency metrics are less relevant. The absolute values of the ECT, DAL, and HRL metrics will be different at different locations. The values will be lower when calculated closer to the storage, and higher when calculated closer to applications. This is expected behavior. The presented techniques use a change in the value and not the value itself. An ECT value of 500 milliseconds, for instance, is not good or bad itself. But if the value increases to 1000 milliseconds after being stable at 500 milliseconds for a week, this would be considered a 100% increase and it indicates a slowdown in storage access.

Third, DAL is the sum of all of the delays that are caused by storage. The block IO transactions include multiple steps and delays are present at every step. It is important to take the sum of all the delays that are caused by storage in calculating the DAL. For example:

- A read IO transaction over a FC fabric has a delay between a command (CMND) and a first data frame (referred to as CTFDT or Command to First Data Time) and between the last data and an RSP frame (referred to as DTRT or Data to RSP Time). The DAL value for the read transactions must be the sum of the CTFDT and the DTRT. Additionally, this logic can be extended to calculate the delay between two adjacent data frames.
- A write IO transaction over a FC fabric has a delay between a CMND and a Transfer Ready frame (referred to as CTFDT) and the last data and an RSP frame (referred to as DTRT). A write transaction has one or more sequences of Transfer ready and data frames. The sum of all the delays must be calculated when multiple instances of Transfer ready are involved.

Fourth, HRL is the sum of all the delays that are caused by the host. The block IO transactions include multiple steps and delays are present at every step. It is important to take the sum of all of the delays that are caused by the host in calculating the HRL. For example, a write IO transaction over a FC fabric has a delay between a Transfer ready and a data frame (referred to as RDYTDT or Transfer Ready to Data Time). A write transaction has one or more sequences of Transfer ready and data frames. The sum of all of the delays must be calculated when multiple instances of Transfer ready are involved. Additionally, this logic can be extended to calculate the delay between two adjacent data frames also.

Figure 3, below, illustrates aspects of the different calculations that were described in the above narrative.

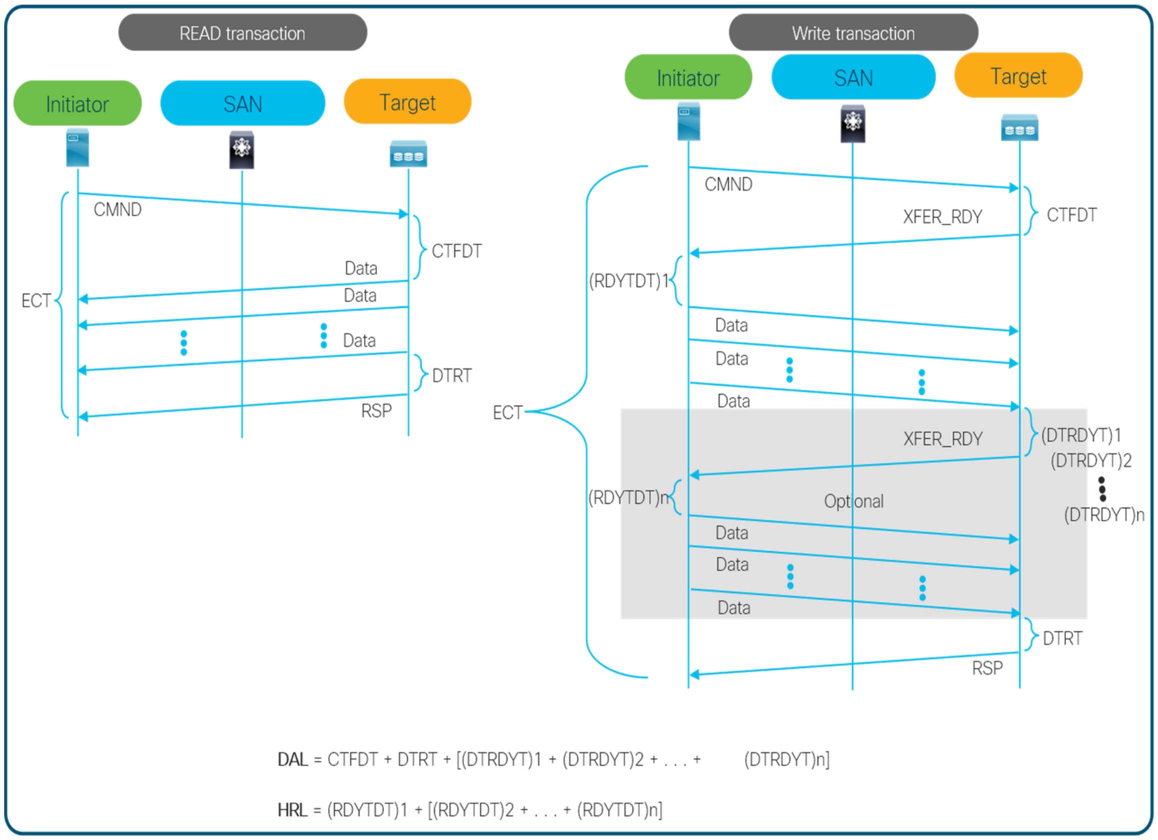

Figure 3: Exemplary Metric Calculations

It is important to note that although the above examples considered FC transport, the same logic will apply to any other transport carrying block storage traffic. Additionally, various of the techniques presented herein, as described and illustrated above, may be automated on a data platform for proactive alerting functionality.

The advantages that may arise from employing aspects of the techniques presented herein may include, for example, that:

- The techniques do not require error counters. They work using existing latency metrics.
- The techniques work even if no errors are observed between applications and storage. They are specifically designed for slow-but-working or sick-but-not-dead situations.
- The techniques work with or without SAN analytics and remain agnostic to the approach of metrics collection.

- The techniques work with any transport carrying block storage traffic (e.g., FC, FCoE, iSCSI, RoCE, and/or NVMe/TCP).

In summary, techniques have been presented herein that support a data-driven and algorithmic approach to pinpoint the exact root-cause (e.g., a host, a SAN, or a storage array) of a storage slowdown. The presented techniques work even when no obvious errors are present (e.g., the storage access is sick-but-not-dead). The presented techniques leverage the latency metrics ECT, DAL, and HRL to pinpoint the exact root-cause of the storage slowdown in a distributed block storage architecture using Small Computer System Interface (SCSI) and NVMe over any transport (e.g., FC, FCoE, iSCSI, NVMe/TCP, RoCE, etc.).