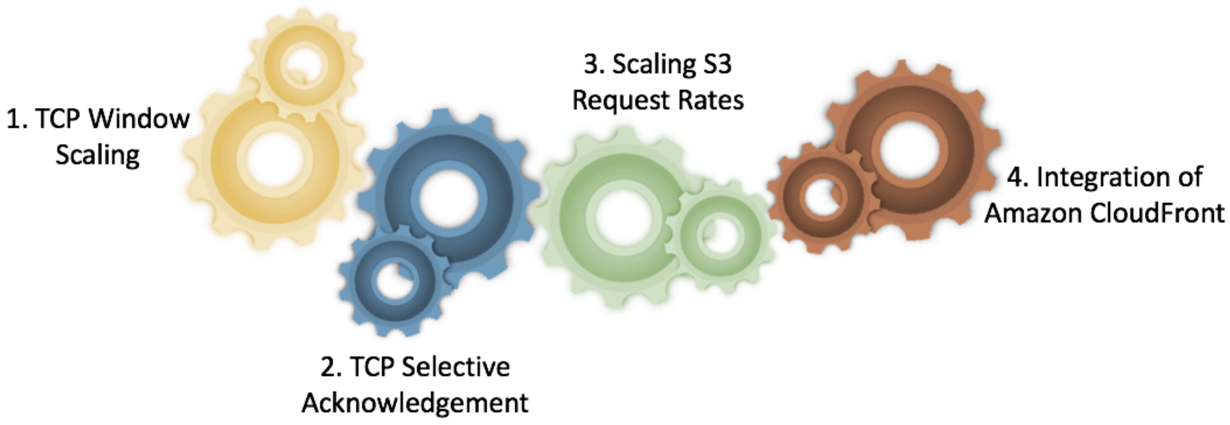
How to Optimize S3 Performance

Whether it's small objects or massive data sets, organizations need to be able to safely store and retrieve a variety of data types.. Amazon S3 object storage offers secure storage for a wide scope of data types in a highly available and resilient environment.

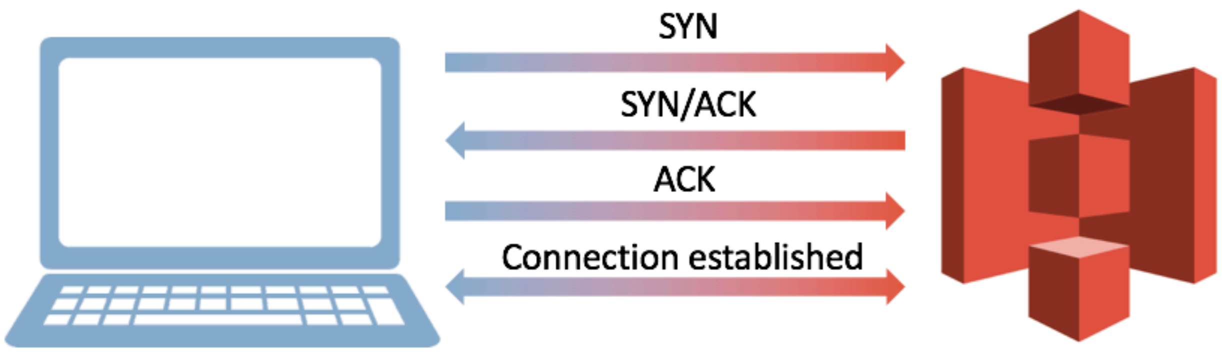
Your S3 objects are likely being read and accessed by your applications, other AWS services, and end users, but are they optimized for maximum performance? Optimizing your usage of S3 makes it possible to achieve faster transfer rate speeds and also reduce costs at the same time, leading to better performance from an efficiency and finance perspective. In this post, I will share some of the techniques and best practices that you can apply to optimize Amazon S3 performance.


1. TCP Window Scaling
2. TCP Selective Acknowledgement
3. Scaling S3 Request Rates
4. Integration of Amazon CloudFront

This method can be used to enhance network throughput performance. By using a window scale to modify the header within the TCP packet, you're able to send a larger amount of data (above the default 64KB) in a single segment. This action isn't specific only to Amazon S3. Because it operates at the protocol level, you can perform window scaling on your client when connecting to any other server using this protocol. Learn more in the RFC-1323 documentation from Internet Engineering Task Force (IETF).

When TCP establishes a connection between a source and destination, a three-way handshake takes place which originates from the source (client). Here's how it works. Before your client can upload an object to S3 a connection to S3 servers must be created. The client will send a TCP packet with a specified TCP window scale factor in the header. This initial TCP request is known as a SYN request, and it's part one of the three-way handshake. In part two, S3 will receive this request and respond with a SYN/ACK message back to the client with its supported window scale factor. Finally, an ACK message acknowledging the response is sent back to the S3 server. On completion of this three-way handshake, a connection is established and data can be sent between the client and S3.

By increasing the window size with a scale factor (window scaling) it enables you to send larger quantities of data in a single segment and therefore more data at a faster rate.



How to Optimize S3 Performance

2. TCP Selective Acknowledgement (SACK)
Sometimes multiple packets can be lost when using TCP. Understanding which packets have been lost within a TCP window can be difficult to determine. While it's possible to resend all of the packets, some of these may have already been received by the receiver. TCP selective acknowledgement (SACK) offers a more effective alternative by notifying the sender only in the case of failed packets within that window. As a result, the sender has to resend only the failed packets.

The request for using SACK must be initiated by the sender (the source client) within the connection established during the SYN phase of the handshake. This option is known as SACK-permitted. Learn more about how to use and implement SACK within the RFC-2018 documentation from IETF.

3. Scaling S3 Request Rates
On top of TCP Scaling and TCP SACK communications, S3 itself is already well optimized for very high throughput. In July 2018, AWS announced significant improvements to request rate performance. Prior to this announcement, AWS recommended that users randomize prefixes within buckets to help optimize performance--this is no longer required. Now, you can apply multiple prefixes within buckets to improve request rate performance.

You are now able to achieve 3,500 PUT/POST/DELETE requests per second along with 5,500 GET requests. These limitations are based on a single prefix, however there is no limit to the number of prefixes that can be used within an S3 bucket. As a result, if you had 20 prefixes you could reach 70,000 PUT/POST/DELETE and 110,000 GET requests per second within the same bucket.

S3 storage operates across a flat structure. This means that there is no hierarchical level of folder structures, you simply have a bucket and ALL objects are stored in a flat address space within that bucket. You may create folders and store objects within that folder, but these are not hierarchical, they are simply prefixes to the object, which help make the object unique. For example, consider that you have the following the data objects within a single bucket: ***Presentation/Meeting.ppt***, ***Project/Plan.pdf***, and ***Stuart.jpg***.

In this scenario, the 'Presentation' folder acts as a prefix to identify the object and this path name is known as the object key. Similarly, with the 'Project' folder, again this is the prefix to the object. The folder 'Stuart.jpg' does not have a prefix and therefore can be found within the root of the bucket itself.

4. Integration of Amazon CloudFront
Another method to improve optimization is to incorporate Amazon S3 with Amazon CloudFront. This works particularly well if the main request to your S3 data is a GET request. Amazon CloudFront is AWS's content delivery network that speeds up distribution of your static and dynamic content through its worldwide network of edge locations.

Normally when a user requests content from S3 (GET request), the request is routed to the S3 service and corresponding servers to return that content. However, if you're using CloudFront in front of S3, CloudFront can cache commonly requested objects. Therefore, the GET request from the user is then routed to the closest edge location that provides the lowest latency to deliver the best performance and return the cached object. This also helps reduce your AWS S3 costs by reducing the number of GET requests to your buckets.

These have been just some of the different options available to help you optimize Amazon S3 performance when working with S3 objects.

Learn more in our Working with Amazon CloudFront course, and get hands-on practice in our lab, Configuring a Static Website with S3 and CloudFront.

Stuart Scott
AWS Content Lead

@Stuart_A_Scott
https://uk.linkedin.com/in/stuartanthonyscott