Demystifying and mitigating TCP stalls at the server side

Ming zhi Yu and Zaina Hamid

EECS 589 Paper Review

Full Reference

Zhou, J. *et al.*, "Demystifying and mitigating TCP stalls at the server side," *Proc. of ACM CoNEXT'15,* 2015.

Summary

In this paper, the authors present a framework, named TAPO, which can be used to diagnose the causes of TCP stalls and propose a mechanism, named S-RTO, to mitigate stalls caused by timeout retransmissions. To determine the cause of TCP stalls, a decision tree is used with inputs being parameters derived from packet-level traces obtained on the server side. By running TAPO on several production traces, TCP stalls are classified into three categories, namely, server-side stalls, client-side stalls, and network-side stalls. While there are multiple causes of stall within each category, the authors focus on stalls caused by timeout retransmission in the network-side stalls category based on the assumption that timeout retransmissions are the most significant contributor for stall time in all traces tested by TAPO. According to the analysis presented in the paper, a primary root cause of timeout retransmission stalls is that TCP is unable to perform a fast retransmit (as opposed to a time-out retransmit) in a timely manner when it could have. Based on this analysis, the proposed mechanism performs fast retransmit more aggressively in order to mitigate timeout retransmission stalls. Evaluations for web search and cloud storage services show that S-RTO is comparatively effective in reducing flow latency substantially.

Highlights

There are several aspects that we like about this paper. First of all, we consider the idea of using a decision tree to accurately determine the cause of a stall as both novel and fundamental to subsequent design and analysis of the algorithm proposed. As we know from basic TCP understanding, the method that TCP uses to provide reliable data transfer as well as congestion control is complicated, which makes identifying the cause of TCP's performance issues a difficult yet extremely important problem. TCP performance optimization has been an area of deep interest due to heavy reliance of varied applications on it. With a decision tree, the complicated process of determining the cause of a stall is made methodical, as it contributes a structured and unified view of all possible classification outcomes, which not only makes it easy for readers to understand how various parameters affect the decision making process but also helps to expedite it. For example, the first step in deciding the cause of a stall is to examine the position of the packet within the file transfer; if the first packet after the stall corresponds to the beginning of a file being transmitted, the framework determines the cause to be 'data unavailable', which obviates the need to examine rest of the parameters extracted from the trace.

In terms of the decision tree, we also appreciate the fine granularity of decision that it is able to reach. For example, in examining the impact of a small sender window on stalls, the authors do not stop when the conclusion that '*in_flight* is small' is reached. Instead, the authors go two steps further by first examining whether the small sender windows is caused by a small congestion window size or a small receiver window size. Once it is determined that a small receiver window size is the cause, the authors continue to look at the initial *rwnd* to see if it is too small to start with.

One of the benefits of performing a fine-grain analysis is that it enables people to identify causes of stalls comprehensively. As mentioned earlier, the way TCP provides reliable data transfer and congestion/flow controls is complicated. Causes to a stall reside not only on the sender side but also on the receiver side. In fact, it is not until reading this paper do we realize the fact that a flow receiver's behavior could also result in a stall. Had the authors not performed such a fine-grained analysis on the root-causes of stalls, our knowledge on various attributing factors would be shallow and accurately pinpointing a fix or a contributing factor would be challenging. Having a comprehensive view and understanding of different causes, and their proportionate values, makes it easier and provides a platform for researchers to continue working further in this specific area of further improvement. For example, although the authors of the paper do not cover techniques used to alleviate client-side stalls in detail, the identification of zero receive window and multiple objects being requested via a single connection as root causes of client-side stalls could definitely serve as a design guide for client side application developers, enabling them to take precautionary measures to avoid potential client side stalls (i.e. the simplest thing is to not make the receive window so small).

Another aspect that we like about the paper and the overall methodology, is that the authors rely on statistics/data obtained during framework experimentation to narrow research scope, and reason out the same with corresponding graphs and ball-park corresponding contributing factors for respective services. As mentioned in the paper, network side stalls are the only category that can be addressed by TCP and hence the authors focus on designing an algorithm to mitigate them. However, because causes leading to network side stalls are not unique and developing a universal solution is difficult, it is more reasonable to develop an algorithm that tackles the predominant aspects of the problem. Without extensive knowledge on causes that are responsible for the majority of network-side stalls, subsequent research on developing a new algorithm to mitigate the stalls could be headed in a wrong direction. For example, the authors call out that network-side stalls can be categorized into timeout retransmission and packet delay. If they would not consider retransmission delay as the most significant contributor for stall time, they could possibly come up with a completely different algorithm which focuses on packet delay stalls instead. By making use of statistics obtained from their analysis framework, the authors can validate their assumptions and narrow the scope of research down to an area that is most likely to lead to a big improvement.

In terms of the design of S-RTO, we appreciate the idea of using a separate timer for each flow to realize a more aggressive retransmission. As mentioned in the paper, another approach that could also result in a more aggressive retransmission is to reduce the value of RTO such that a timeout retransmission can happen earlier. However, this simple fix could possibly have a negative impact because whenever a

timeout retransmission happens, sender's congestion window is reset to 1 MSS and starts to ramp up slowly, which results in an underutilization of available bandwidth. In other words, adopting the simple fix would possibly introduce a new problem. Hence, instead of tuning existing parameters in TCP's congestion control algorithm, the authors introduce a separate timer, which not only enables a timely retransmission but also avoids messing up TCP's existing congestion control mechanism. Additionally, comparison of performance improvements in the case of TLP and S-RTO relative to the native Linux while categorizing different quantiles is an efficient way of summing up comprehensive results in a compact style.

Improvements and possible extensions

As mentioned, the overall description of stalls, and their impact in different areas of cloud storage, security software download and web search has been described in great detail, with metrics governing most of the reasoning. The clarity of reasoning across the above services, and the categorization of stalls based on server side, client side, and the network, with corresponding explanation is also extremely helpful. However, there isn't substantial emphasis placed on the design architecture of the tool, and the thought process involved in coming up with the decision tree structure, except basing off of parameters gathered off the server side. Also Client idle stalls and Resource constraint stalls as depicted in the percentage of stalls vs. time table contribute a significant chunk towards the overall percentage calculation, however these are missing from the decision tree depiction and the flow with respect to these stalls is slightly ambiguous.

Another point that came across was while categorising in Table 5, is that stalls that could meet several types are short circuited to fall in the double retransmission category if applicable. We understand the degradation caused due to retransmission stalls and the reasoning behind it, however choosing the order in which stalls are examined remains slightly ambiguous though it ensures better categorisation and classification into a single type.

Possible future work could be comparative analysis of performance after effective mitigation of f-double stalls, tail retransmission stalls in Open state, and Continuous loss stalls as briefly mentioned towards the conclusion of the respective stalls. S-RTO analyses the mitigation of tail retransmission and cwnd retransmission stalls, however leaving out f-double stalls, and the latency comparison shows results comparing Linux, TLP and S-RTO.

Additional shortcomings that were not addressed in detail are:

- Throughput of large flows did not show a significant improvement
- Evaluation of TLP and S-RTO against Linux Kernel has been done in specific for Web Search and Cloud storage, while excluding Security software download.
- Disadvantages of using S-RTO other than the fact that some stalls will be missed out are not highlighted with respect to the bigger picture.
- The consequences due to additional retransmissions generated due to aggressive retransmissions has been left as future work.