Contribution ID: **41** Type: **Talk**

Managing Bufferbloat in Storage Systems

Wednesday, 2 December 2020 15:45 (30 minutes)

Scalable storage servers consist of multiple parts that communicate asynchronously via queues. There is usually a frontend that queues access requests from storage clients and uses one or more threads to forward queued requests to a backend. The backend queues these forwarded requests and batches them to efficiently use storage devices it manages. Storage servers can have multiple kinds of backends with different design assumptions about their underlying storage device technologies. Requests are scheduled in the frontend to ensure different levels of service for different classes of requests. For example, requests that are generated by data scrubbers working in the background generally have a lower priority than requests from an application. A common solution to the above problem is to move request scheduling from the frontend to the backend. For various reasons that is not always practical. The scope of the proposed project is to have the scheduler reside in the frontend and to explore designs for backends to dynamically control the admission of requests depending on continually changing workloads and storage device technologies.

Scheduling in the frontend and batching in the backend work best if there are enough requests in their respective queues. This raises the question: what is enough for the frontend and for the backend? If there are too few requests in the frontend but more than enough requests in the backend, the system might work well in terms of overall throughput but might poorly enforce scheduling objectives. If there are too few requests in the backend, then overall throughput and latency suffer no matter the scheduling objectives. If, however, the backend has the ability to admit just enough requests from the frontend but not more, throughput and latency of the backend is likely satisfactory. If there is enough work overall, the frontend has enough requests to meet scheduling objectives. How many requests are just enough for the backend?

In this talk I will give an overview of an ongoing research project at the UC Santa Cruz Center for Research in Open Source Software (cross.ucsc.edu) to reframe this question as a bufferbloat mitigation problem using algorithms similar to the ones used for bufferbloat in networking.

Student?

No

Supervisor name

Supervisor email

Primary authors: MALTZAHN, Carlos (University of California, Santa Cruz); MIRVAKILI, Esmaeil (University of California, Santa Cruz)

Presenter: MALTZAHN, Carlos (University of California, Santa Cruz)

Session Classification: HPC

Track Classification: Storage and IO