Optimizing Fsync Performance with Dynamic Queue Depth Adaptation

Min Ji Kim, Daejun Park, and Dongkun Shin
College of Information & Communication Engineering, Sungkyunkwan University, Suwon, Korea
E-mail : dongkun@skku.edu

Recent storage devices such as eMMC and SSD support the command queueing [1] in order to improve the storage I/O bandwidth. The command queueing allows multiple read/write requests to be pending. Since the multi-channel and multi-way architectures of eMMC and SSD can handle multiple requests simultaneously, the command queueing is indispensable technique for them. However, the command queueing can be harmful to the latency of fsync system call, the latency of which is critical to application responsiveness. All pending I/O requests should be completed before the fsync system call. We investigated the fsync latencies under different queue depths. As shown in Fig 1, the fsync latency increases as the queue depth increases since the fsync latency is affected by the number of queued I/O requests which have arrived before the fsync-related requests. However, the I/O bandwidth reaches the maximum value when the queue depth is equal to 4, and there is no significant changes when the queue depth is larger than 4. Therefore, too large number of queue depths will be detrimental to fsync latency without any improvement on I/O bandwidth. We propose a queue depth adaptation technique, which reduces the queue depth before user application sends fsync calls as shown in Fig 2. We profiled the file system behavior, and found several file types for which the fsync calls are frequently used. For example, SQLite DB and .xml files are flushed into the storage by the fsync calls. The proposed technique monitors the opened files, and reduces the queue depth if there is a high probability of fsync calls in order to provide fast responsiveness.

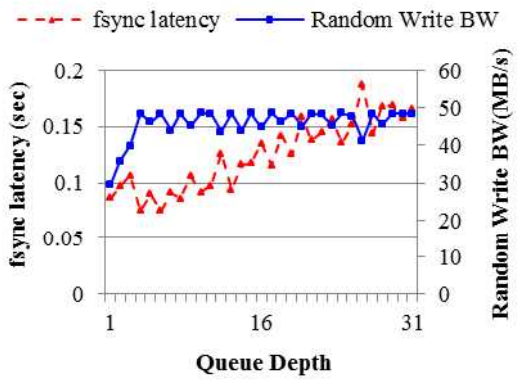

Fig 1 Increment of fsync latency w/ random writes

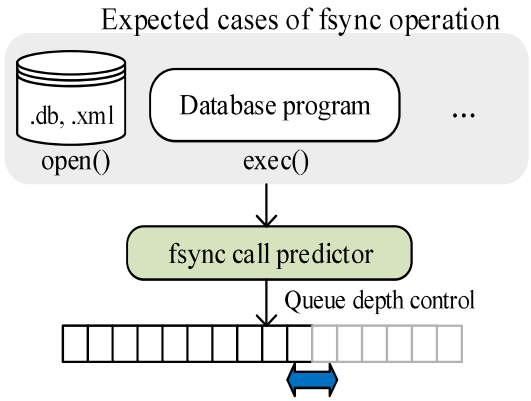

Fig 2 Queue depth Adaptation technique

[1] Dees, Brian. "Native command queuing-advanced performance in desktop storage." Potentials, IEEE 24.4 (2005): 4-7.