Paper Review:
"*An Efficient Memory-Mapped Key-Value Store for Flash Storage*"

Seri Lee
Seoul National University
sally20921@snu.ac.kr

***Abstract*—**

***Index Terms*—**

I. Introduction

We first observe two significant sources of overhead in state-of-the-art key-value stores are: (a)The use of compaction in LSM-Trees that constantly perform merging and sorting of large data segments and (b) the use of an I/O cache to access devices, which incurs overhead even for data that reside in memory.

II. Memory-Mapped I/O

Most key-value stores and other systems that handle data use explicit I/O to access storage devices or files with read/write system call. In many cases, they also employ a user-space cache as part of the application to minimize accesses to storage devices. However, even the use of an application user-level cache incurs significant overhead in the common path.

The use of memory-mapped I/O in Kreon reduces CPU overhead related to the I/O cache in many ways. Memory-mapped I/O uses a single address space for both memory and storage, which eliminates the need for pointer translation between memory and storage address spaces, and therefore, the need to serialize and deserialize data when transferring between the two address spaces.

[1]
[2]