# POCSD Project – 3 Venkata Bharathula UFID: 8737-9584

**Persistent Hierrachical File System:**

**Design:**

**Persistence using MongoDB:**

The existing remote hierrachical file system is interfaced with MongoDB to achieve persistence. To save the file data into the MongoDB a key value pair is used, where key is the file ID and value is the defaultdict(bytes). These key value pairs are stored in the db  as files collection.

When files are created, key value pairs are created in the mongodb for the file id and then  when writes to the files happen, key value pairs are written into the MongoDB file system.

Reads are handled by obtaining the data from the MongoDB file system and then returning the appropriate chunk based on the offset.

**Cached Implementation:**

For the cached implementation, I have used a custom cache implementation which follows the LRU model for replacing the objects in the cache. When a file is created, read, written LRU cache is set with the current file in memory.

Writes to the DB happen when the current

Writes are implemented to both the cache and the MongoDB. Reads to the DB occur only in the case of a cache Miss.

**How to Run:**

```
./persistentFSMongo.py <mountpoint>
./cachedPersistentFS.py <mountpoint>
./cachedPersistentFS_new.py <mountpoint>
```

**Tests Performed:**

Tested for copy of files larger than 10MB on to the file system and observed the performance and also tested for all general file system operations.

**Performance Comparison between cached and Non cached File System:**

Read from the file system is improved from 30KBps to 500KBps by using the Cached file system. In writes there is no significant improvement if writes are always carried out to the mongodb instance.

Tried a different approach to save the writes in cache and write them when cache is being purged and in the end. The same is implemented in the  cachedPersistentFS_new.py.