POCS OP 3

6.10.2023

Name : Yonghao Zou

SCIPER : 3 7 0 9 2 3

SCIPER 3 7 0 9 2 3

Question:

Twizzler enables applications to directly access NVM. Explain how the Twizzler designers applied the "Make the common case fast, and the rare one merely correct" principle in this context. Make sure you state clearly what the assumed common/rare cases are, and how Twizzler handles them. Will Twizzler always outperform a conventional kernel, where each access to storage is mediated by the kernel?

(Write your answer inside the box below. Anything outside the box will be ignored. Your answer can have at most 10 sentences.)

The first common case is reading/writing the NVM and Twizzler made it fast by removing kernel from access path via a global object space and self-contained object. Object is accessed by object id and access control is verified, ~~in~~ ~~when accessed~~, by hardware. The corresbonding rare one is creating/deleting object, where kernel is involved for allocating object id and changing related data structure like view.

The second common case for persistent pointer is that ~~most~~ accessing data inside same ~~one~~ object which does not require slow pointer translation but by bit-or operation. While the counter part is accessing other objects which need pointer translation by FOT table.

Twizzler does not always outperform a conventional kernel when an application has many creating/deleting operations and few cross-object operations like the KV example in the experiment.

(You can use this page for scratch notes. It will be discarded prior to grading.)

common case: Read / write

rare: create/delete

pesistent pointer: offset in 1 object
points to diff objects.