Memory Persistency

For this paper, it introduces the memory persistency and related multithread conceptions & manipulations. For memory consistency, “In multithreaded programs, some processes might be doing read operations (search) and others might be doing an update operation (insert/remove)”. It “allow programmers to reason about the visible order of loads and stores among threads”. For memory persistency, it guarantees that all threads would have the same content and results. It is really important because for threads we need to let them unify to make programming and debugging easier. For sequential consistency, it guarantees that the executing sequence would be predictable and ordered. Because of the attribute for non-volatile memory, it is also possible to recover the status of memory immediately after crush (For example, halt, blue screen, etc.)

For discussion, the professor mentioned that for volatile memory, it needs checkpoint & save to other spaces (HDD, etc) to backup & recover the state of the memory. For non-volatile memory, it just restore what it saved for the last time and restore it. Also, for this paper, we do not talk about the burden for programmer to make such multithread approach. What we care about is just making the process of multithread predictable. Also, another challenge for this is different performance for different kinds of non-volatile memory. The write rate is different from read rate for NVM and different approach for it makes it very different for different NVMs. It is also the challenge for memory persistency.