Persistent Lock-Free Algorithms

Overview:

Nonvolatile memory (NVM) is the next-generation memory which is able to replace the currently wide-used DRAM in few decades. Compared with DRAM, NVRAM has advantages in power consumption and recovery from power failure, since for NVRAM, there is no need to regularly refresh for contents retaining, and contents would not be lost even power failure happens. In other word, NVRAM is a persistent storage similar to disks but much faster and more fine-grain.

Along with the advantages, however, tons of traditional algorithms designed for DRAM currently will not work in NVRAM. Instead, within the field of NVRAM, a problem named persistency needs to be carefully considered. Persistency stands for whether contents in memory is persistent and consistent, even power failure occurs arbitrarily. Once memory does not lose its contents after power failures, persistency is crucial for the correctness of algorithms.

Suppose variables x and y are variables to be stored. First we assign x to be 1, and we compute y as x+1. Now we have x=1 and y=2. But what if power failure occurs in these operations? If we do not pay enough attention to the algorithm, it is possible that when power is down, though x=1 and y=2 were in caches, but only y=2 was written back to the memory while x=1 not. Thus after recovery, we get y=2, but x in undecided value, which violates the equation y=x+1. Programmers may have if(y==2) statement and using x to do operations implying that x=1.

Potential Benefits:

NVRAM is a good candidate for portable devices and high-performance computing centers. The former requires power consumption to be low, and possibility of recovery from any crashes is attractive for the latter. As a result, in decades NVRAM will be widely used in lots of cases. Alongside, the persistent algorithms would be applied in any software which runs in computing devices with NVRAM equipped.

Persistency can be intuitively guaranteed by flushing cache every time after a meta operation finishes, which is named as flush-as-you-go, but the performance of this method must be unattractive, since every operation have to wait for its previous operations to finish multiple flushing. As a result, how to reach persistency with high performance is a valuable topic, which ensures whether NVRAM and the algorithm are practical enough to be wide-spread.
Furthermore, persistency is also beneficial for transactional memory, which is useful for database system. Previous work [1] mentions a method different from the intuitive one. The method does periodical write-back to ensure the persistency, which means, within a period, the operations can be treated as a transaction so that they would either all success or all fail, without interruption to each other.

**Research Design:**

By now, there are some papers describing persistent algorithms that are basic, such as queue and hash map. But that is far from sufficiency. Based on previous works, our research goal is to build and implement persistent algorithms for the future use. It consists with three parts.

To begin with, the research would start with implementing and generalizing the persistent hash map named Dalí [1]. As described before, Dalí is a periodically persistent hash map retaining multiple versions of contents to make sure the persistency. Based on Dalí, we can generalize to other algorithms such as queue/list and stack.

Next, we would step into ordered maps, such as skip list and search tree. Search tree is another important data structure in computer science, though expected to be much difficult to implement with tricks.

Finally, after efficient algorithms are carefully designed and analyzed, we plan to implement and package all these persistent algorithms into a library, with carefully designed APIs, like Standard Template Library in C++.

**Reference**


