**8-2 Short Paper: Deadlock Avoidance**

**Christopher Gray**

**IT-600**

**Southern New Hampshire University**

**Professor Michael Gallaher**

**Introduction:**

In the course of Top Secret, Inc. (TSI) implementing its enhanced back-office operating system, Top Secret, Inc. (TSI) has encountered a critical issue pertaining to the lack of mechanisms for ordered acquisition of synchronization objects. This has resulted in periodic lockups within its web servers, impacting the company's ability to provide consistent web services. The dominant problem appears to stem from the absence of a structured approach for acquiring synchronization locks, leading to deadlock scenarios where the web server process gets trapped in a call to pthread\_mutex\_lock(). Despite other processes appearing active, the unavailability of the web server process during these occurrences disrupts customer access and transactions during peak usage. This paper aims to propose a suitable deadlock avoidance approach tailored to TSI's web server application in light of the absence of ordered mutex acquisition.

The core issue lies in the lack of a predefined order for acquiring synchronization locks, causing threads to encounter deadlocks when contending for these resources. In cases of high demand, this mismanagement of locks halts the progress of the web server process, rendering it inaccessible to users and affecting the company's revenue by obstructing customer interactions. To address TSI's deadlock issue and ensure the orderly acquisition and release of synchronization locks, the Two-Phase Locking (2PL) approach is a suitable solution. This technique systematically divides the process of lock management into two distinct phases, mitigating the chances of encountering deadlocks (blog.purestorage.com, 2022).

During the growing phase, threads are required to request synchronization locks in a predetermined order. This is in stark contrast to conventional mutex locks, which lack this order-based acquisition (w3.cs.jmu.edu, n.d.). Threads attempting to acquire locks already held by other threads will be temporarily blocked, effectively preventing indefinite delays and the formation of circular wait patterns. After completing their respective critical sections, threads enter the shrinking phase. Here, all acquired locks are released simultaneously in a single atomic action. This release mechanism ensures that no other thread can attempt to acquire these locks before the current thread has fully relinquished them, reducing potential conflicts.

The adoption of the Two-Phase Locking (2PL) approach offers a range of advantages for addressing deadlock issues within Top Secret, Inc.'s (TSI) web server application. First and foremost, by imposing a predetermined sequence for acquiring locks and ensuring the simultaneous release of all acquired locks, 2PL effectively eliminates the potential for circular waits, which are a core driver of deadlock scenarios. This heightened level of control over lock acquisition and release enhances the stability of the system by preventing threads from becoming entangled in unbreakable impasses. Additionally, the transparent and consistent nature of the lock acquisition and release process under the 2PL framework fosters a more predictable system behavior. This predictability, in turn, simplifies the debugging and analysis of the system's operation, allowing for more efficient identification and resolution of potential issues. Furthermore, the inherent blocking and wake-up mechanisms intrinsic to the 2PL approach contribute to fairness in lock acquisition among concurrent threads. By temporarily blocking threads attempting to acquire locks held by others, the system ensures that no thread is perpetually deprived of access to the required resources, thus promoting equitable resource distribution. However, there will be some challenges in implementation. Implementing the Two-Phase Locking approach in TSI's web server application necessitates modifying the existing lock management mechanism to adhere to the prescribed order of lock acquisition. Coordinating the growing and shrinking phases will involve careful synchronization mechanisms to prevent race conditions. This might require reengineering sections of the application's codebase.

The recommended Two-Phase Locking approach offers a potent solution to TSI's deadlock concerns within its web server application. By imposing an ordered lock acquisition process and promoting fairness in resource sharing, 2PL can effectively prevent deadlocks, enhancing the stability of the system and mitigating customer dissatisfaction stemming from unavailability during peak usage periods. While the implementation demands effort, the resulting benefits in terms of system reliability and customer satisfaction make it a justifiable endeavor for TSI's continued success.

References:

Storage, P. (2022). Why You Should Use Strict 2PL over 2PL. Retrieved 26 August 2023, from https://blog.purestorage.com/purely-informational/why-you-should-use-strict-2pl-over-2pl/#:~:text=The%20two%2Dphase%20locking%20(2PL,change%20or%20update%20that%20data.

7.3. Locks — Computer Systems Fundamentals. (2023). Retrieved 26 August 2023, from https://w3.cs.jmu.edu/kirkpams/OpenCSF/Books/csf/html/Locks.html#:~:text=A%20lock%20is%20also%20called,first%20thread%20completes%20its%20work.