APPLICATION OF SWIFT CONCURRENCY IN MOBILE APPLICATIONS FOR OPTIMIZING LARGE-SCALE DATA PROCESSING

PhD, Dmytro Hlavchev, PhD student Pavlo Borysov, Kharkiv

Processing large data volumes in modern mobile apps presents significant performance challenges. Traditional iOS asynchronous tools like Grand Central Dispatch (GCD) suffer from limitations such as thread explosion, architectural complexity, and main UI thread blocking. These issues can degrade performance by 40–50% and negatively impact user experience [1].

The use of Swift Concurrency technology is proposed as an alternative approach to optimising large-scale data processing. The technological foundation of the solution is the Swift Concurrency framework with async/await syntax and the actor system, which provides structured concurrent programming. The key advantages are the Cooperative Thread Pool with a fixed number of threads equal to the number of CPU cores, preventing thread explosion and optimising system resource utilisation. An architectural solution has been developed based on Structured Concurrency principles, including four modules: data segmentation (automatic partitioning of arrays into optimal segments), task dispatching (withTaskGroup for parallel execution), data processing (async transformation algorithms), and result aggregation (async sequences for merging). Parallel processing algorithms were implemented using TaskGroup and async let to efficiently distribute workload among available CPU cores without blocking the main UI thread [2].

Experimental studies on arrays ranging from 10^3 to 10^6 elements confirmed the high efficiency of Swift Concurrency for mobile application optimisation. The following results were obtained: a 35 – 60% performance increase compared to GCD when processing arrays larger than 10^4 elements, a 25% reduction in memory consumption due to automatic task lifecycle management, and a 45% improvement in UI response time thanks to non-blocking data processing. Swift Concurrency demonstrates advantages in application stability by eliminating race condition and memory leak issues typical of traditional approaches. The research results confirm the feasibility of using Swift Concurrency as a core technology for data-intensive mobile applications.

References: 1. A. Kärrby, 'Evaluating Swift concurrency on the iOS platform: A performance analysis of the task-based concurrency model in Swift 5.5', Dissertation, 2022. **2.** Apple Inc. Swift Concurrency Documentation. Developer Documentation, 2025. URL: https://developer.apple.com/documentation/swift/concurrency/.