Aurendil: Rethinking the Graph Query Engine

While graph database management systems (GDBMS) have matured significantly, their underlying runtime architectures often inherit relational execution models that exhibit pathological performance on certain classes of query. Specifically, operations like triangle counting—a core primitive for graph analytics—suffer from intermediate result explosion and redundant computation. This talk investigates the root causes of these inefficiencies, identifying that volcano-style iteration and standard join-based plans fail to exploit the structural symmetries and recurring sub-computations inherent in graph topologies.

To address these challenges, we present a novel architectural shift that integrates established techniques from optimizing compilers and managed runtime systems into the graph database runtime. We describe the motivation and construction of Aurendil, an experimental graph runtime system that treats graph queries as a program that can be aggressively optimized. Specifically, we borrow three key techniques from the field of programming languages: redundancy elimination, which identifies redundant computations that occur whilst executing the query; strength reduction, which replaces high-cost operators with semantically equivalent but lower-cost operators; and specialized execution, where common or complex plans can be recognized and replaced with non-relational operators specifically implemented for graph workloads. This approach creates a novel optimization environment that affords Aurendil a performance envelope that far exceeds the current generation of GDBMS.

We evaluate this design through a targeted case study on triangle counting, comparing Aurendil against a range of state-of-the-art graph database runtimes including both transactional and analytic systems. Our results demonstrate that Aurendil's single-threaded execution achieves a median performance improvement of 18.1X over the next best performing graph database and 34.9X over the existing Neo4j Pipelined runtime. Notably, even when compared against state-of-the-art multi-threaded runtimes, Aurendil achieves a 2.03X speedup over the Neo4j Parallel Runtime. We show how this performance gain is fundamentally achieved via the systematic elimination of redundant database accesses. The consequent reduction in hardware resource utilization affords Aurendil the ability to operate on larger volumes of data without requiring additional hardware; which has a profound effect on the total cost of ownership (TCO) in cloud environments. We estimate that an Aurendil-like architecture could lead to a 3X to 35X reduction in cloud compute costs, providing a sustainable and high-performance path forward for large-scale GDBMS in the cloud era.