Aurendil Project

Graph databases are an increasingly popular method for storing data because they allow developers to model and query complex domains. For this reason, graph databases are ideally suited to many tasks that support future AI applications, such as building knowledge graphs and analyzing complex networks.

In this talk, we will introduce Aurendil, a project exploring how we construct next-generation query runtimes for cloud-based graph databases. Importantly, Aurendil departs from the traditional construction of query languages and looks to instead lean on concepts from programming language runtimes, such as the Java Virtual Machine, to permit queries to be dynamically compiled and optimized over successive executions. Such a shift in thinking will open up our integration options with modern cloud hardware, allowing us to use GPGPUs, FPGAs, and Smart NICS to drive up scalability and performance.

Over the lifetime of the Aurendil project, there are three significant challenges to address: increasing the efficiency of graph database technology to lower operational costs and improve performance; developing technologies to scale graph databases so that they can cope with the exponential growth of users' data; and addressing the evolving landscape of computer hardware.

Our starting point is to revisit how Graph Pattern Matching (GPM) works in contemporary graph databases at a fundamental level and whether we can improve performance, scalability, and stability. We will show that specific GPM queries encounter problems such as excessive memory usage or pathologically bad performance due to the volume of intermediate results that they have or materialize or their inability to optimize specific types of GPM query.

Over the past six months, we have been constructing an optimizing framework that drastically improves the hardware efficiency of GPM matching in Neo4j. Our alternate execution model for GPM and use of a more programming language like intermediate representation (IR) help

to support a broader range of dynamic optimization, enabling us to perform optimization at a much finer granularity than before. For example, we can show how to reduce complex GPM queries into semantically equivalent but simpler forms that take advantage of schema metadata to eliminate the need to access the database completely. Our results show that our Aurendil prototype can perform the same GPM queries with a dramatically reduced memory footprint, which means we can tackle larger problems with less hardware. Finally, we will outline the next steps for the Aurendil project.