RAVINDRA, PADMASHREE. Optimizing RDF Analytical Queries on MapReduce. (Under the direction of Kemafor Anyanwu.)

The broadened use of Semantic Web technologies to enable data integration solutions across domains has increased the amount of semi-structured data on the Web represented using the Resource Description Framework (RDF). This has led to a growing interest to support analytics over semantically integrated RDF data warehouses, such as analysis of patient and drug profile data in life sciences for more effective clinical trial recruiting, and analysis of people-places data by e-government decision makers to improve citizen facilitation. In order to support large scale RDF analytics, it is crucial to investigate how to leverage parallel processing systems such as MapReduce and extended systems such as Apache Hadoop, Pig, and Hive.

An RDF analytical query consists of, (i) graph pattern matching to compute query-relevant subgraphs, (ii) grouping desired attributes, and (iii) aggregating values. In general, graph pattern matching translates to several join operations due to the fine-grained nature of the RDF data model. Many analytical queries involve multiple groupings and aggregations over slightly different graph patterns, further increasing the number of join operations. Evaluating such join-intensive workloads on existing platforms results in lengthy execution workflows. The challenge with such lengthy workflows is the I/O and network transfer costs due to the intermediate data produced across multiple map-reduce phases. Such costs can be significant for workloads that produce large intermediate results. Consequently, it is critical to develop techniques that enable more nimble execution of such workflows.

In this dissertation, I present a holistic approach to minimize the I/O and network transfer costs while processing RDF analytical queries on MapReduce. Given that RDF analytical queries often involve repeated computations over slightly different graph patterns, query plans that enable shared execution of common subpatterns are likely to compile into efficient MapReduce execution plans. To achieve this, I propose the following three optimization techniques that exploit sharing opportunities while evaluating RDF analytical queries.

- First, I propose an algebraic optimization approach that enables shared execution of overlapping graph patterns, thus eliminating the multiple phases of I/O and materialization involved in evaluation of multiple graph patterns in an RDF analytical query. A decoupling of the grouping and aggregation definitions is used to enable sharing of scans and computations across the graph pattern matching phases, as well as the grouping-aggregation phases. Such a rewriting results in an aggressive reduction in the length of execution workflows.
Second, I propose an algebraic optimization of basic graph pattern queries using an alternative data model and algebra called the Nested TripleGroup Data Model and Algebra (NTGA). The NTGA query plans enable concurrent computation of star-shaped join sub-patterns in a query, as opposed to existing systems that require a separate map-reduce cycle for each star subpattern. Thus, by enabling **sharing of scans and computations across multiple star subpatterns**, the NTGA query plans result in reduced numbers of map-reduce cycles.

Third, I propose strategies for efficient management of intermediate results while evaluating graph pattern queries with multi-valued and unbound properties. For such queries, normalized representations of intermediate results using relational join operations introduce redundancies in results. To mitigate the effects of such redundancies, the NTGA’s nested data model and lazy **unpacking** strategies enable **sharing of data references, scans, and computations**, thus reducing the footprint of intermediate results.

We extended Apache Pig’s computational infrastructure to integrate the NTGA-based data model and operators along with the optimization strategies. Empirical evaluation on real-world and synthetic benchmark datasets demonstrate that the NTGA-based query plans result in shortened execution workflows with reduced footprint of intermediate results, thus minimizing the I/O and network transfers while processing RDF analytical queries on MapReduce.