A Two-Tiered Index Architecture for Scalable RDF Processing
Sairam Gurajada and Martin Theobald {gurajada,mtb}@mpi-inf.mpg.de

Introduction
Who acted in a “tv-show” as well as in a “movie”? Vincent Donofrio is one answer

Motivation
How to scale RDF System?
• Over 30 Billion triples in the linked data cloud
• Distributed Approach

Challenges
• Minimize inter-node communication by effective partitioning and replication approaches
• Parallel query processing and efficient load balancing

Existing Approaches
• Distribute the triples by applying hashing on Subject or Object Inefficient to answer path queries, but Efficient load balancing
• Use graph clustering algorithms on large RDF graph and distribute the triples with 1 or 2-hop replication Poor load balancing, but efficient in processing path queries?

Problem Statement
• A Partitioner P partitions the RDF graph G into (G1, G2, G3, ..., Gk)
• A query Q which requires multiple (m per say) partitions to answer is split in to (Q1, Q2, ..., Qm)
• Let Ri be the set of results returned by executing Qi independently on partition Gi
• During join operation, - Rj results are shipped to partition Rk (Rj < Rk) and joined to form result set Rj (<< Rk and << Rj)
- Tuples Communicated: Rj

Our Idea is to have a “Two-Tiered Index architecture”
Tier 1 : Super graph index contains the summary of original graph
Tier 2 : Original graph index

Current work
• How to build Super Graph (Summary graph) from RDF graph?
• Indexing approaches for Super Graph and regular RDF graph
• Can the state of the art single site RDF systems can be used locally at each slave?
• How to design a replicator for two-tiered index architecture

Sparql to SQL Convertor
SQL Query Processor
RDF Triple Store – Relational Model

References:
3) Lei Zou et al. sStore: answering SPARQL queries via subgraph matching, VLDB 2011