Towards A Petascale RDF Data Processing Framework Using Pig and Hadoop

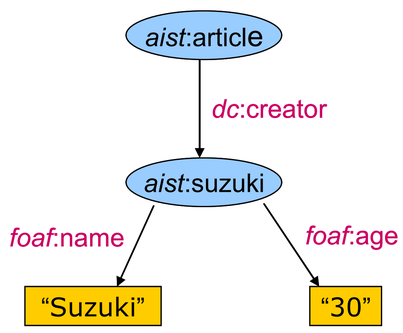
Yusuke Tanimura, Akiyoshi Matono, Steven Lynden, Isao Kojima
National Institute of Advanced Industrial Science and Technology (AIST), Japan
{yusuke.tanimura, a.matono, steven.lynden}@aist.go.jp, kojima@ni.aist.go.jp

Background

What is RDF?

■ RDF stands for Resource Description Framework, which is a W3C standard for describing Web documents and resources from the real world - people, organizations, things and so on.

■ RDF data consists of a set of triples (subject, predicate and object) and forms a graph, which can represent knowledge networks with RDF Schema and OWL (Web Ontology Language).



Example of RDF triples

1)	***aist*:article**	***dc*:creator**	***aist*:suzuki** .
2)	***aist*:suzuki**	***foaf*:name**	“Suzuki” .
3)	***aist*:suzuki**	***foaf*:age**	“30” .

The size and number of RDF repositories is increasing.

Current situation (MB ~ GB):

- W3C SWEO (Semantic Web Education and Outreach) Linking Open Data community reported in May, 2009, that the community interlinked over 4.7 billion RDF triples with about 142 million links.
- The DBpedia knowledge base describes more than 2.6 million things using 274 millions’ RDF triples and the size is about 67GB.

Near future:

- The number of the RDF triples and the number of the RDF repositories will increase more. Pure RDF data repository will be GB ~ TB scale.
- The contents repository which stores data and its metadata (RDF data) will be TB ~ PB scale.

System requirements for RDF data processing

■ Sufficient computing power for RDF queries (graph pattern matching)

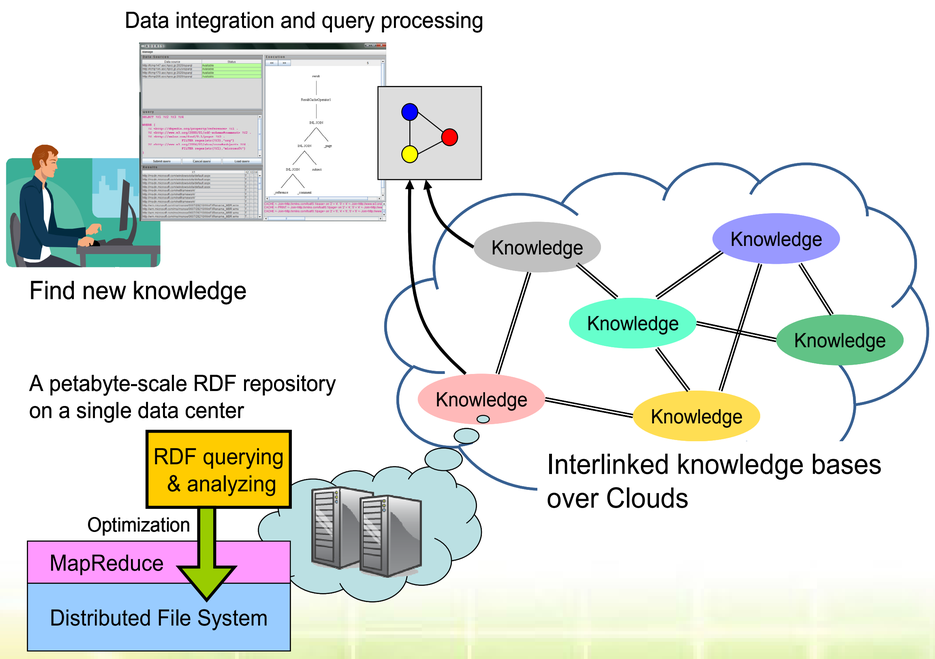
■ Flexible user interface for RDF specific queries or rule-based inference

➧ They are not achieved by existing relational databases.

Project Goal

Develop core technologies for building a scalable and distributed knowledge base using RDF:

- A user side data integration system, which joins data from multiple RDF repositories in an efficient manner
- A data provider side single scalable RDF repository system

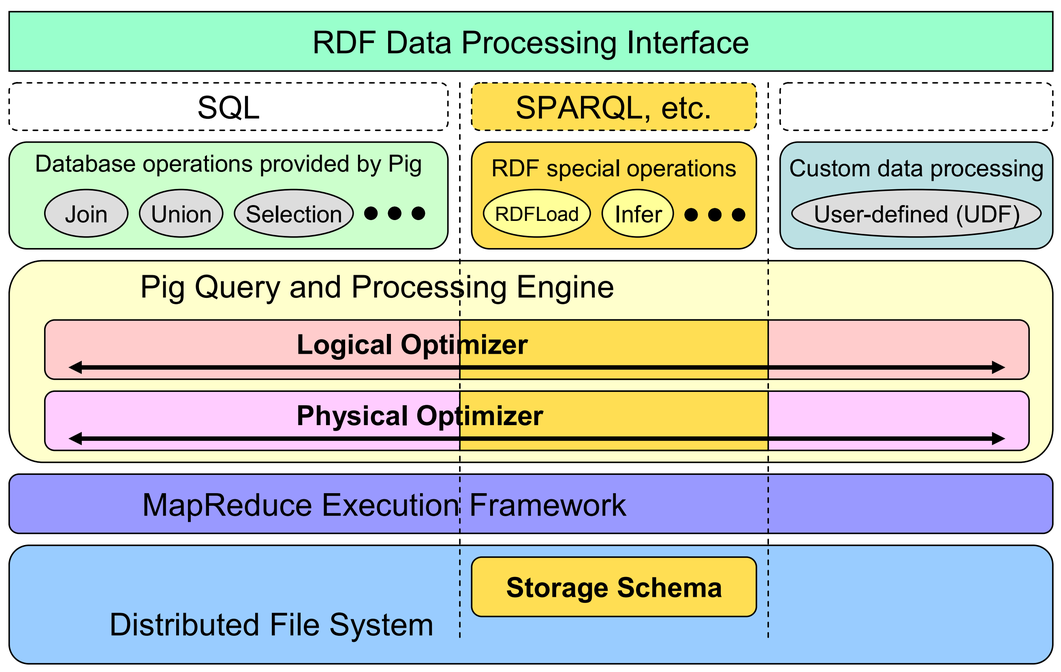


Proposed Scalable and Flexible RDF Data Processing Framework on A Single Site

Scalable architecture: A Pig and Hadoop-based system (Distributed File System and MapReduce model)

RDF extensions on top of the general data processing framework (Pig/Hadoop):

- The extension involves storage schema and query optimization to make query execution faster.
- The extension provides an interface of the inference operations as Pig Latin language, with internal query optimization.



Efficient RDF Data Access

■ Defining storage schema

✓ A basic file structure of the RDF triples, intermediate results and indices

➢ Based on Hadoop’s MapFile format

✓ Data partitioning (File partitioning and distribution on DFS)

➢ Vertical Partitioning

✓Structure of indices including inferred data

■ Implementing an RDF data loader using the schema

The system provides RDFLoader() as a built-in function of Pig. The function takes a SPARQL filtering statement as an argument, and omits reading unnecessary data from DFS. The LOAD command using the function looks like below.

```
a = LOAD ‘DBpedia’ RDFLoader( ‘?predicate = http://www.w3.org/ ....’ );
```

Reasoning Support

The system provides the RDF-INFER command, which is able to infer new RDF triples by processing a set of user-defined rules. The command looks like below.

```
b = RDF-INFER USING RDFRuleBase( ‘myrule-1’ );
```

The system internally optimizes the RDF-INFER operation. For example of the transitive closure, the operation may involve multi-stages self-joins. We plan to implement adaptive techniques; choosing the best join algorithm implemented with MapReduce, using join indices or not, and so on. The optimization is expected to reduce large amount of overheads of the MapReduce execution.

A part of this work is supported by Strategic Information and Communications R&D Promotion Programme (SCOPE) of the Japanese Ministry of Internal Affairs.