

MA-INF 4223 - Lab Distributed Big Data Analytics

SANSA: Scalable Semantic Analytics Stack

Dr. Hajira Jabeen, Gezim Sejdiu, Denis Lukovnikov

Summer Semester 2019



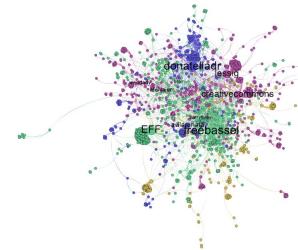
Lesson objectives

- ❖ After completing this lesson, you should be able to:
 - Understand the usage of SANSA
 - List and understand the libraries offered by SANSA



SANSA: Motivation

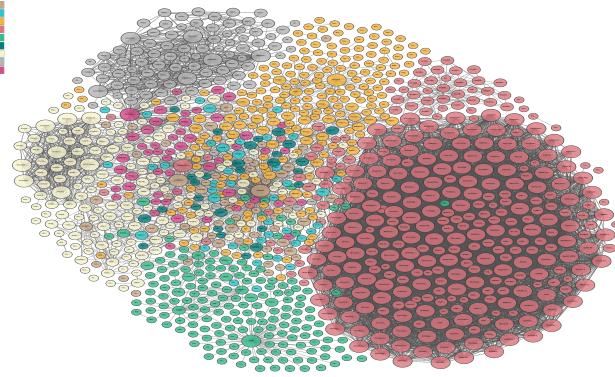
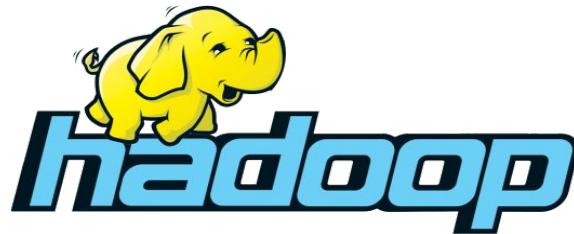
- ❖ Abundant machine readable structured information is available (e.g. in RDF)
 - Across SCs, e.g. Life Science Data
 - General: DBpedia, Google knowledge graph
 - Social graphs: Facebook, Twitter
- ❖ Need for scalable querying, inference and machine learning
 - Link prediction
 - Knowledge base completion
 - Predictive analytics





SANSA: Motivation

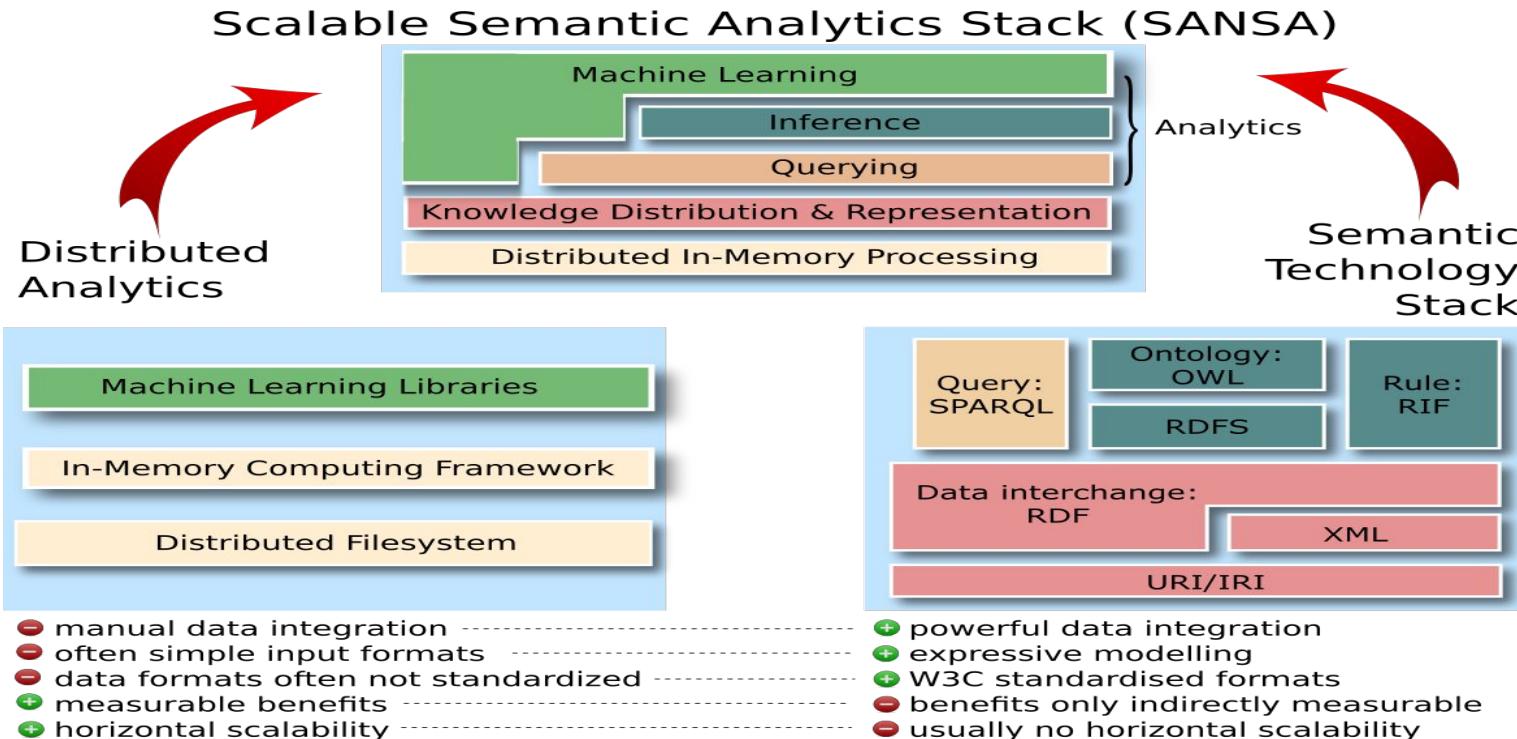
- ❖ Over the last years, the size of the **Semantic Web** has increased and several large-scale datasets were published



Source: LOD-Cloud (<http://lod-cloud.net/>)

- ❖ Now days **hadoop** ecosystem has become a standard for **BigData** applications
- ❖ We use this infrastructure for Semantic Web as well

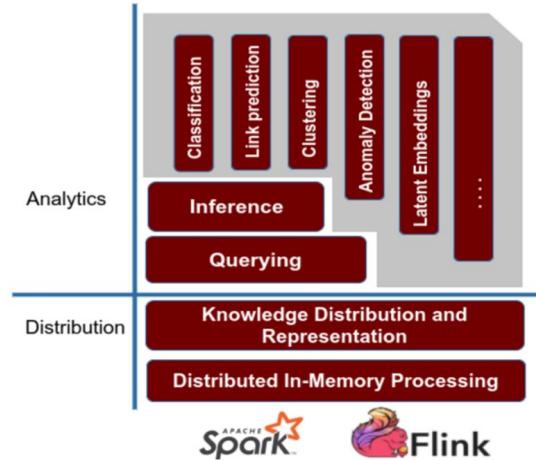
A New Vision Combining Semantic Technologies and Distributed Machine Learning





SANSA Stack

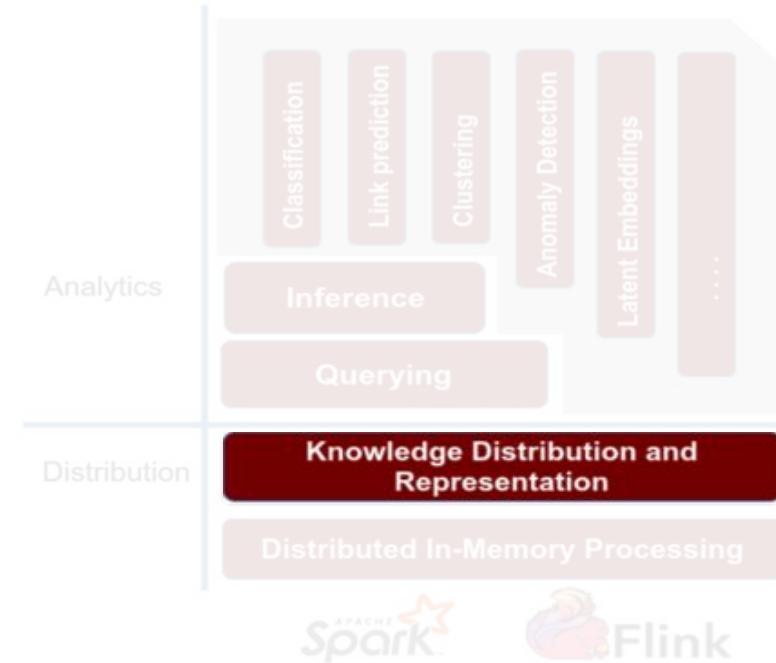
- ❖ It's core is a **processing data flow** engine that provides data distribution, and fault tolerance for distributed computations over RDF large-scale datasets
- ❖ SANSA includes **several libraries** for creating applications:
 - [Read / Write RDF / OWL library](#)
 - [Querying library](#)
 - [Inference library](#)
 - [ML- Machine Learning core library](#)



<http://sansa-stack.net/>



SANSA: Read Write Layer





SANSA: Read Write Layer

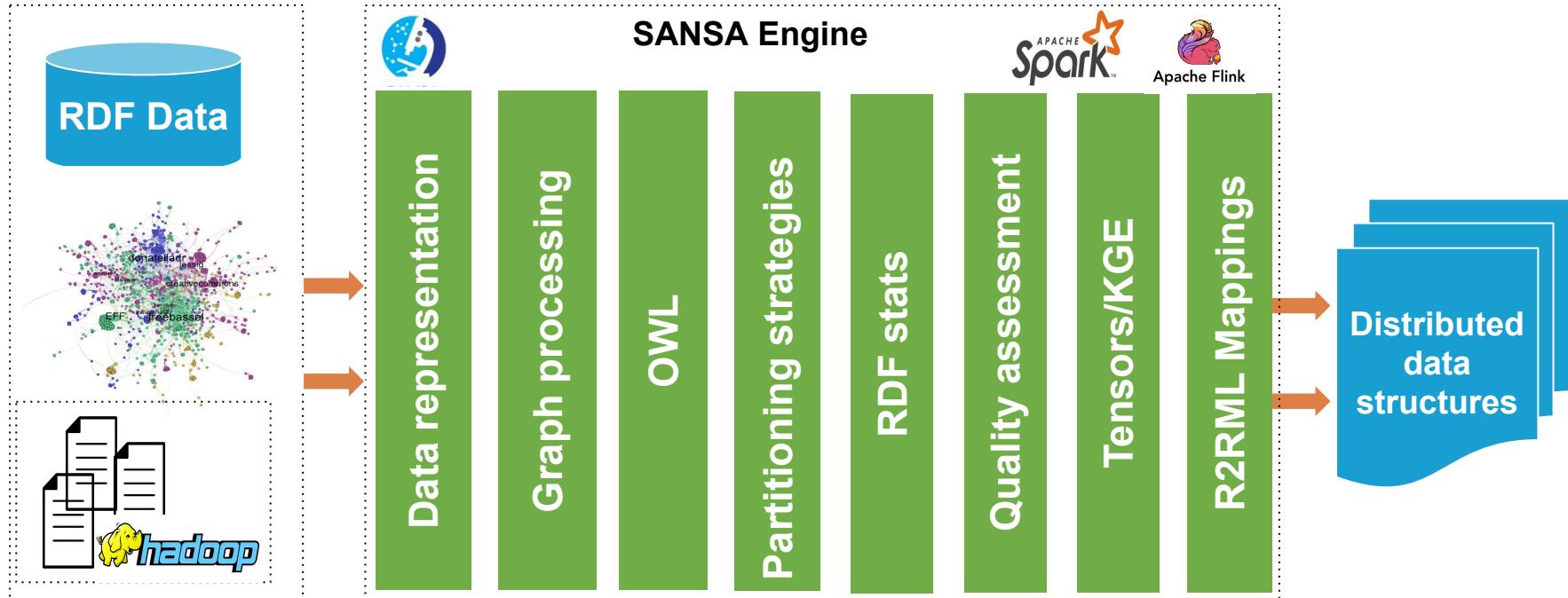
- ❖ Ingest RDF and OWL data in different formats using Jena / OWL API style interfaces
- ❖ Represent data in multiple formats
 - (e.g. RDD, Data Frames, GraphX, Tensors)
- ❖ Allow transformation among these formats
- ❖ Compute dataset statistics and apply functions to URLs, literals, subjects, objects → Distributed LODStats



```
val triples = spark.rdf(Lang.NTRIPLES)(input)
triples.find(None,
Some(NodeFactory.createURI("http://dbpedia.org/ontology/influenced")), None)
val rdf_stats_prop_dist = triples.statsPropertyUsage()
```



SANSA: Read Write Layer features





SANSA: OWL Support

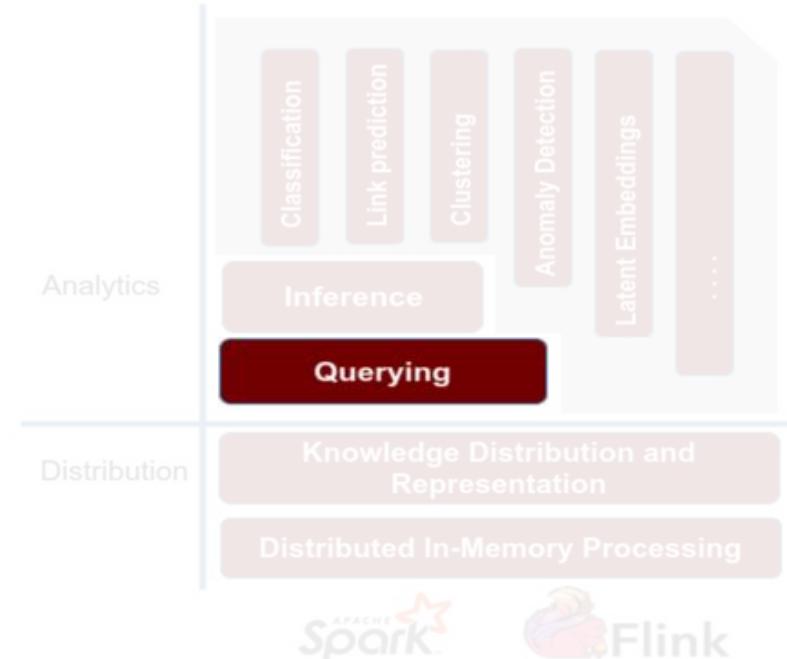
- ❖ Distributed processing of OWL axioms
- ❖ Support for Manchester OWL & functional syntax
- ❖ Derived distributed data structures:
 - E.g. matrix representation of subclass-of axioms to compute its closure via matrix operations



```
val rdd = spark.owl(Syntax.MANCHESTER)("file.owl")
// get all subclass-of axioms
val sco = rdd.filter(_.isInstanceOf[OWLSubClassOfAxiom])
```



SANSA: Query Layer





SANSA: Query Layer

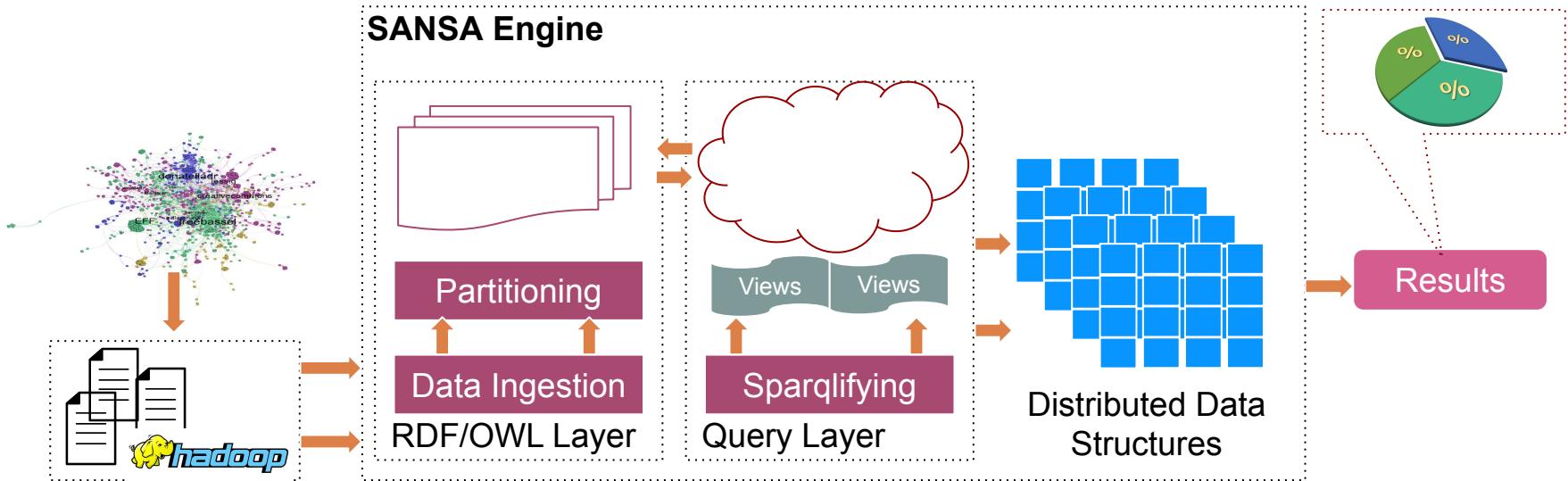
- ❖ To make generic queries efficient and fast using:
 - Intelligent indexing
 - Splitting strategies
 - Distributed Storage
- ❖ SPARQL query engine evaluation
 - (SPARQL-to-SQL approaches, Virtual Views)
- ❖ Provision of W3C SPARQL compliant endpoint



```
val triples = spark.rdf(Lang.NTRIPLES)(input)  
  
val sparqlQuery = "SELECT * WHERE {?s ?p ?o} LIMIT 10"  
val result = triples.sparql(sparqlQuery)
```



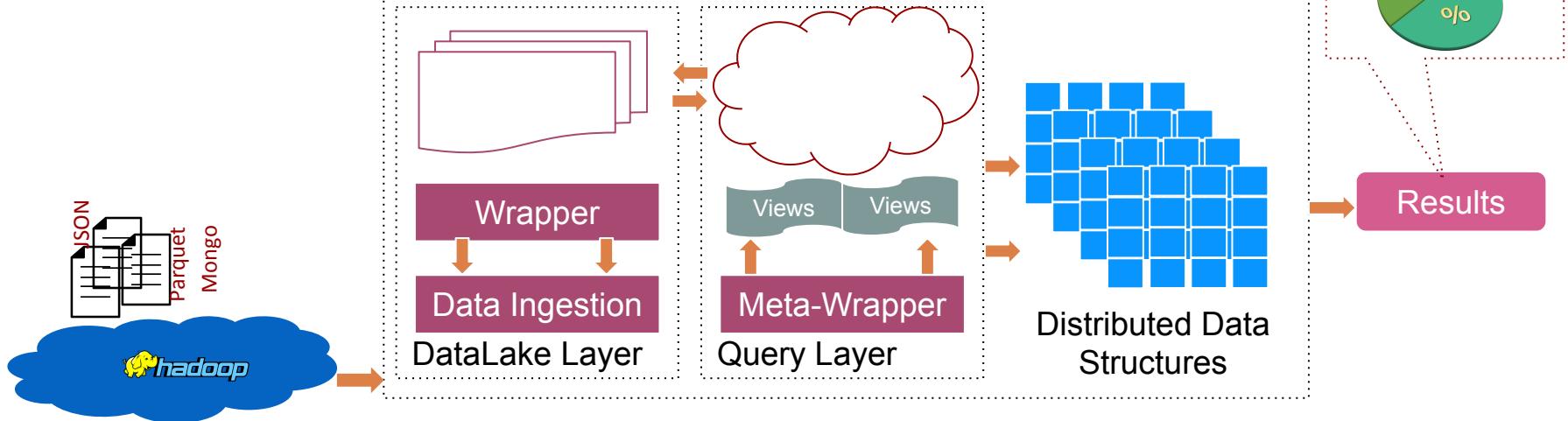
Querying via SPARQL & Partitioning





Querying via Semantic Data Lake

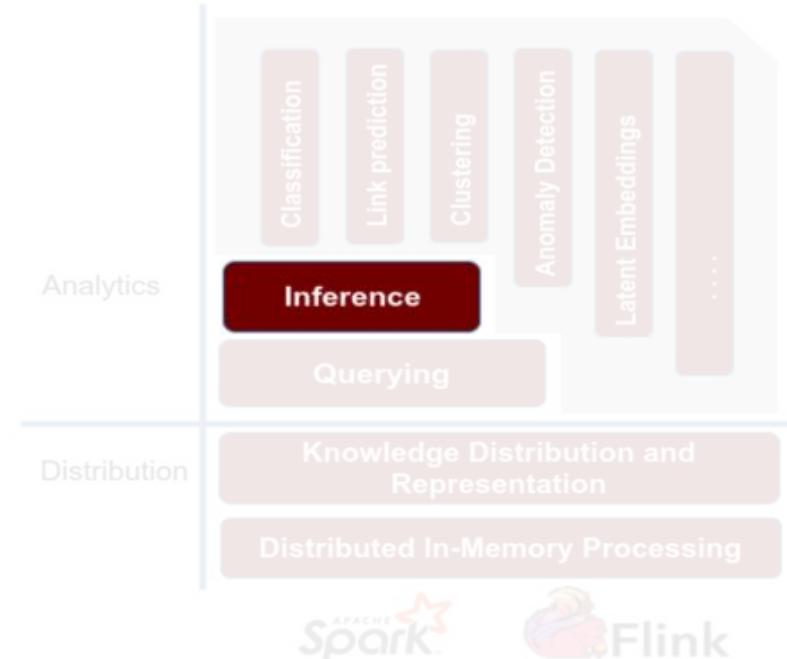
SANSA Engine



```
val result = spark.sparqlDL(query, mappingsFile, configFile)
```



SANSA: Inference Layer

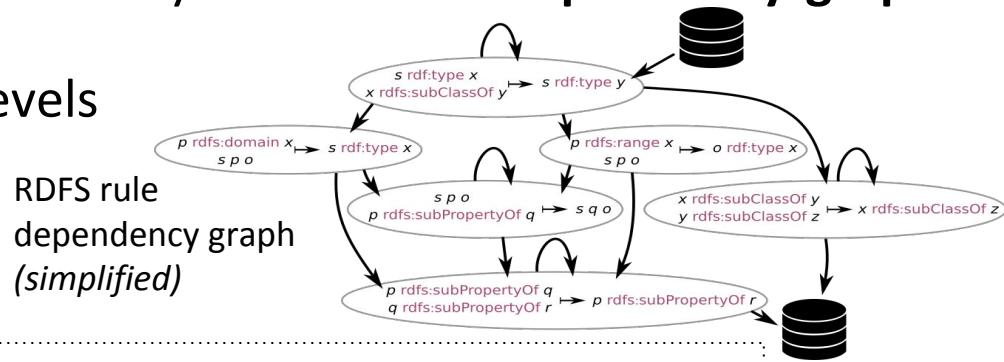




SANSA: Inference Layer



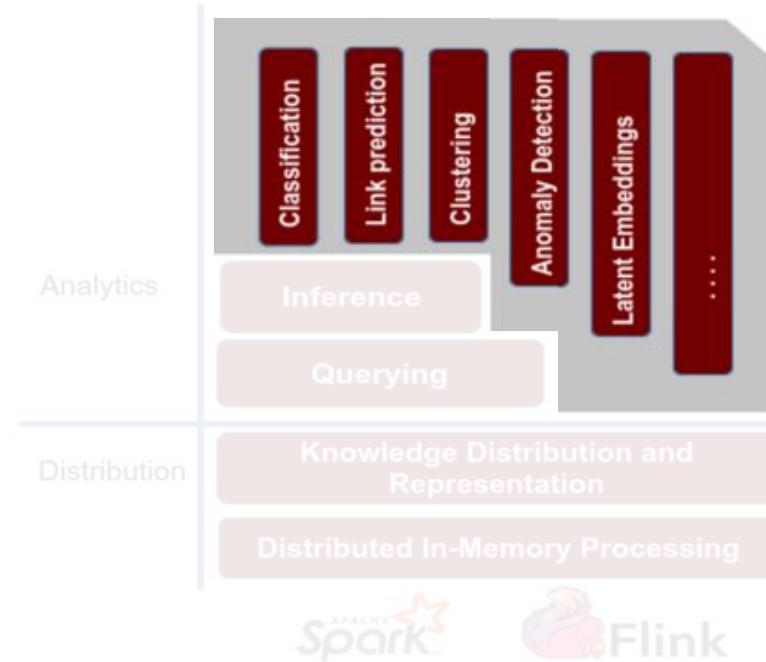
- ❖ W3C Standards for Modelling: RDFS and OWL
- ❖ Parallel in-memory inference via rule-based forward chaining
- ❖ Beyond state of the art: dynamically build a **rule dependency graph** for a rule set
- ❖ → Adjustable performance levels



```
val graph = RDFGraphLoader.loadFromDisk(spark, uri)
val reasoner = new ForwardRuleReasonerOWLHorst(spark.sparkContext)
val inferredGraph = reasoner.apply(graph)
```



SANSA: ML Layer





SANSA: ML Layer

- ❖ Distributed Machine Learning (ML) algorithms that work on RDF data and make use of its structure / semantics
- ❖ Algorithms:
 - Knowledge graph embeddings for e.g. KB completion, link prediction
 - Graph Clustering
 - Power Iteration, BorderFlow, Link based
 - Modularity based clustering
 - Association rule mining (AMIE+ = mining horn rules from RDF data using partial completeness assumption and type constraints)
 - Semantic Decision trees (in progress)
 - Outlier detection





Interactive SANSA in your Browser

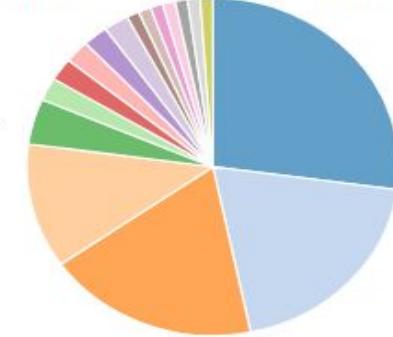
```
val input = "hdfs://namenode:8020/data/rdf.nt"
val triplesRDD = NTripleReader.load(spark, JavaURI.create(input))

val propertyDist = PropertyUsage(triplesRDD, spark).PostProc()
    .map(f => f._1.getLocalName + "\t" + f._2)

println("%table Property Distribution\tFrequency\n " + propertyDist.mkString("\n"))

input: String = hdfs://namenode:8020/data/rdf.nt
triplesRDD: org.apache.spark.rdd.RDD[org.apache.jena.graph.Triple] = MapPartitionsRDD[27] at map at NTripleReader.scala:39
propertyDist: Array[String] = Array(author 25, source 18, description 17, date 11, permission 4, version 2, influenced 2, deathPlace 2, givenName 2, hidetitle 1, wikidata 1, gallery 1, width 1, inline 1, artist 1, year 1)
```

FINISHED ▶ ✖️ 📄 ⚙️



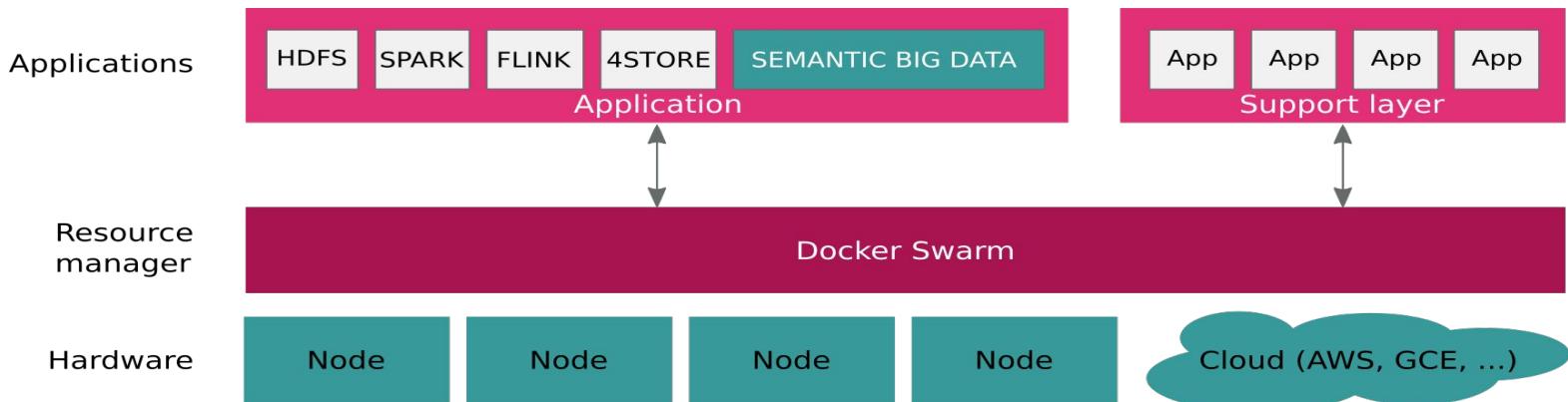
settings ▾

- author ● source ● description ● date ● permission
- gallery ● width ● inline ● artist ● year
- version ● influenced ● otherVersions ● deathPlace
- givenName ● hidetitle ● wikidata



BDE & General Integration

- ❖ SANSA = Scala / Maven Repositories based on Spark / Flink
- ❖ Easy to include both in BDE platform and any Spark / Flink environment





SANSA Planning and Pulse

- ❖ SANSA 0.5 in December 2018, releases every 6 months
- ❖ Apache Open Source License
- ❖ Project activity:
 - Contributors (*at least one commit*): **17**
 - Commits per day: **5.9** - Commits previous year: **2175**
 - Github stars (all repos): **187**

Nov 8, 2015 – Nov 7, 2018

Contributions: **Commits** ▾

Contributions to develop, excluding merge commits





Conclusions and Next steps

- ❖ A generic stack for (big) Linked Data
 - Build on top of a state-of-the-art distributed frameworks (**Spark, Flink**)
- ❖ Out-of-the-box framework for scalable and distributed semantic data analysis combining semantic web and distributed machine learning for (1) querying, (2) inference and (3) analytics of RDF datasets.
- ❖ Next steps
 - Support for SPARQL 1.1 and other partitioning strategies (Query Layer)
 - Backward chaining and better evaluation (Inference Layer)
 - More algorithms and definition of ML pipelines (ML Layer)



References

- [1]. "Linking Open Data cloud diagram 2017, by Andrejs Abele, John P. McCrae, Paul Buitelaar, Anja Jentzsch and Richard Cyganiak. <http://lod-cloud.net/>".
- [2]. "[Distributed Semantic Analytics using the SANSA Stack](#)" by Jens Lehmann, Gezim Sejdiu, Lorenz Bühmann, Patrick Westphal, Claus Stadler, Ivan Ermilov, Simon Bin, Muhammad Saleem, Axel-Cyrille Ngonga Ngomo and Hajira Jabeen in Proceedings of 16th International Semantic Web Conference – Resources Track (ISWC'2017), 2017.
- [3]. "[The Tale of Sansa Spark](#)" by Ivan Ermilov, Jens Lehmann, Gezim Sejdiu, Lorenz Bühmann, Patrick Westphal, Claus Stadler, Simon Bin, Nilesh Chakraborty, Henning Petzka, Muhammad Saleem, Axel-Cyrille Ngonga Ngomo, and Hajira Jabeen in Proceedings of 16th International Semantic Web Conference, Poster & Demos, 2017.
- [4]. "[DistLODStats: Distributed Computation of RDF Dataset Statistics](#)" by Gezim Sejdiu, Ivan Ermilov, Jens Lehmann, and Mohamed Nadjib-Mami in Proceedings of 17th International Semantic Web Conference, 2018.
- [5]. "[Divided we stand out! Forging Cohorts fOr Numeric Outlier Detection in large scale knowledge graphs \(CONOD\)](#)" by Hajira Jabeen; Rajjat Dadwal; Gezim Sejdiu; and Jens Lehmann. In 21st International Conference on Knowledge Engineering and Knowledge Management (EKAW'2018), 2018.

THANK YOU !

< <http://sda.cs.uni-bonn.de/teaching/dbda/> >



Dr. Hajira Jabeen
jabeen@cs.uni-bonn.de
Room 1.062
(Appointment per e-mail)



Gezim Sejdiu
sejdiu@cs.uni-bonn.de
Room 1.068
(Appointment per e-mail)



Denis Lukovnikov
lukovnikov@cs.uni-bonn.de
Room 1.064
(Appointment per e-mail)