SPARTI
Scalable RDF Data Management Using Query-Centric Semantic Partitioning

Amgad Madkour (Purdue)
Walid G. Aref (Purdue)
Ahmed M. Aly (Google)

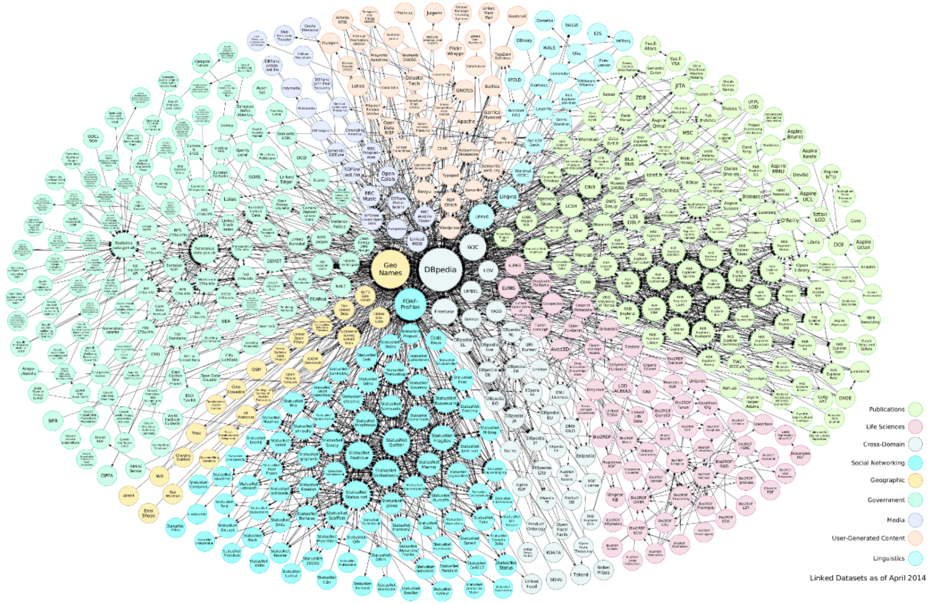
PURDUE UNIVERSITY

Motivation

- The astronomical growth of RDF data raises the need for **scalable** RDF management strategies

- Efficient RDF **data partitioning** can significantly improve the **query performance** over cloud platforms

- **Cloud platforms** provide shared memory, storage, and advanced data processing components that help manage **web-scale RDF**



Problem Definition

- Vertical Partitioning (VP) is a **scalable** partitioning schema that can be used in cloud-based systems
- However, **not all entries** of a VP are part of the **final result**
- The non-matching entries cause computation and communication **overhead**

```
SELECT ?x ?y  WHERE {
   ?x :mention  :Mary .
   ?x :tweet    ?y    . }
```

Join(mention sub, tweet sub)

SUB	OBJ
:John	:Mary
:John	:Mike
:Alex	:Mary

Join(tweet sub, mention sub)

SUB	OBJ
:John	:T1
:Alex	:T4

Reductions

mention

SUB	OBJ
:John	:Mary
:John	:Mike
:Alex	:Mary
:Sally	:Mike
:Mary	:John

tweet

SUB	OBJ
:John	:T1
:Mike	:T2
Mike	:T3
:Alex	:T4

Original Data

Vertical Partitioning Model

Vertical Partitioning [Abadi et al. – VLDB 2009]

- Create **property-bound tables** consisting of two columns
- **Advantages**
  - Inspects the corresponding VP tables only
  - Avoids the tuple-header reading overhead of row-stores when stored in a column store
- **Drawbacks**
  - Some partitions can account for a large portion of the entire graph
    - (i.e. Massive I/O)

Triples Table

Subject	Property	Object
:John	mention	:Mary
:John	mention	:Mike
:John	tweet	:T1
:Mike	tweet	:T2
:Mike	tweet	:T3
:Alex	mention	:Mary
:Alex	tweet	:T4

mention

Subject	Object
:John	:Mary
:Alex	:Mary
:John	:Mike

tweet

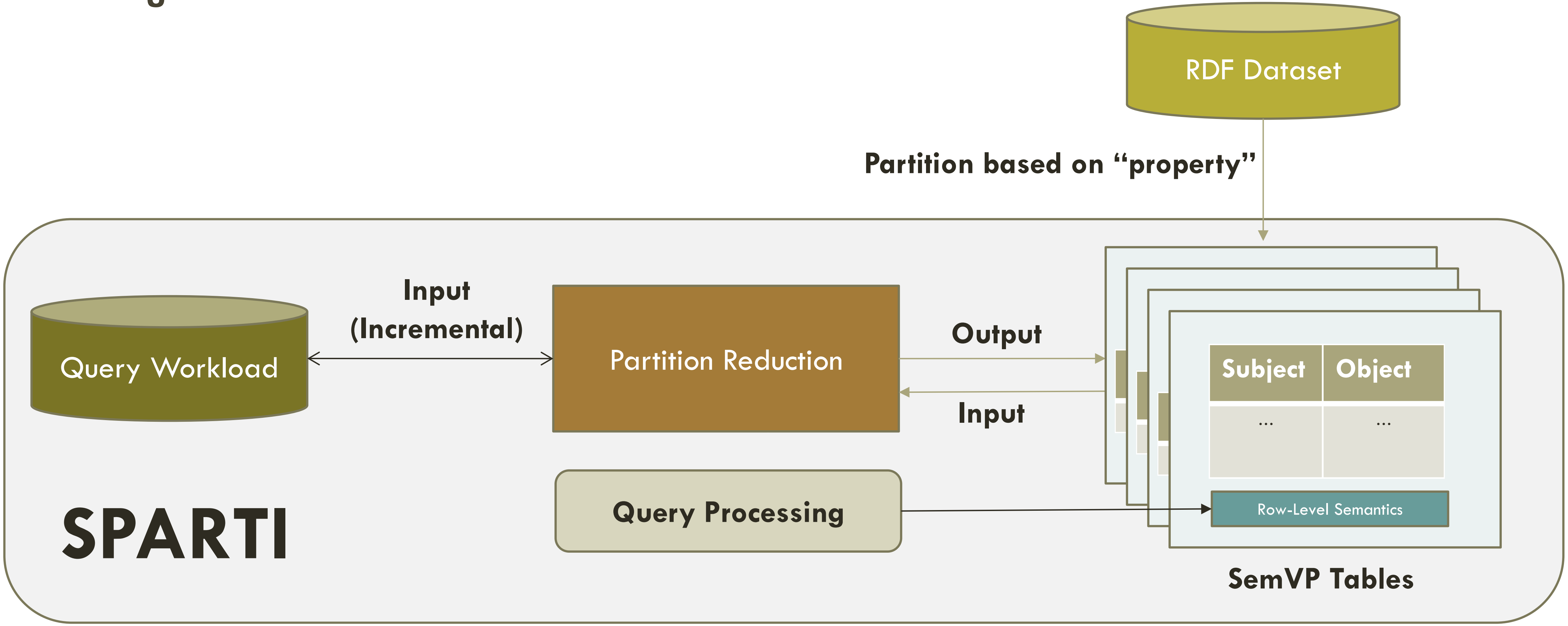
Subject	Object
:John	:T1
:Mike	:T2
:Mike	:T3
:Alex	:T4

Proposal
Overview

- **Three Phases**

1. Property-based Partitioning
2. Partition Reduction
3. Query Processing

Proposal
High-Level Workflow



Proposal
Property-Based Partitioning

- **SemVP** - A relational partitioning schema that extends VP's
  - Partition RDF datasets based on property
  - Represent row-level semantics for every triple via **semantic filters**

- Create **Bloom filters** for every SemVP (subject and object)
  - Used later to compute reduced row-sets for specific query join patterns

Proposal
Partition Reduction

- Identify **related properties** from the query workload for every partition
- Identify **join patterns** in the **query workload**
- Compute a **reduction** using **Bloom join** between **related properties** based on the join patterns
- **Store** the result of the reduction as **semantic filters**

Proposal
Query Processing

- Identify the **properties** and **join patterns** in queries
- **Match** every property and query join pattern with a **SemVP** partition and a **semantic filter**, respectively
- In case a match is found, a **semantic filter** (representing the reduced set of rows) is read to answer a query **instead of** reading the **entire partition** for a property.

Challenges

- How to represent row-level semantics (**semantic filters**)
- How to efficiently compute **reductions** ?
- How to **select semantic filters** that minimize network and disk IO ?
- How to **evolve** based on the query-workload

Semantic Vertical Partitioning (SemVP)

- A relational partitioning schema that extends vertical partitioning with row-level semantics
- Consists of **2 columns**: Subject, Object
- A **Semantic Data Layer (SDATA)** structure that stores triple semantics

Semantic Vertical Partitioning (SemVP)
Example

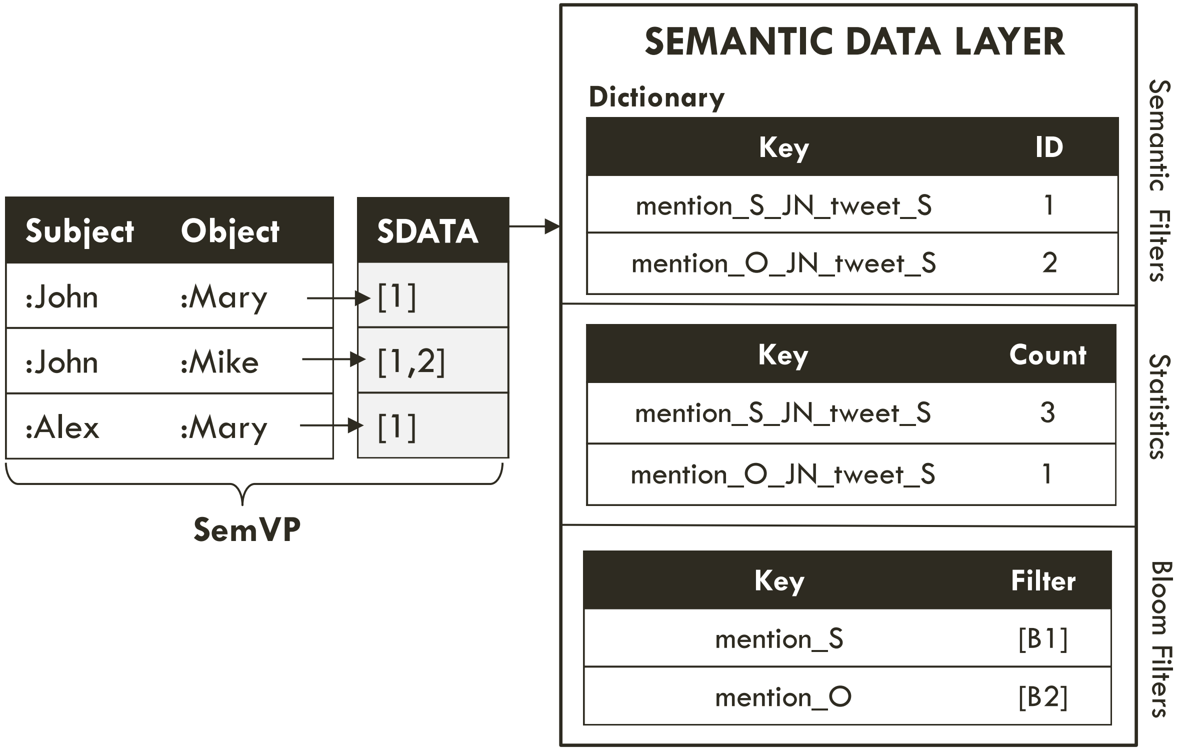


Figure: SemVP representing the mention property

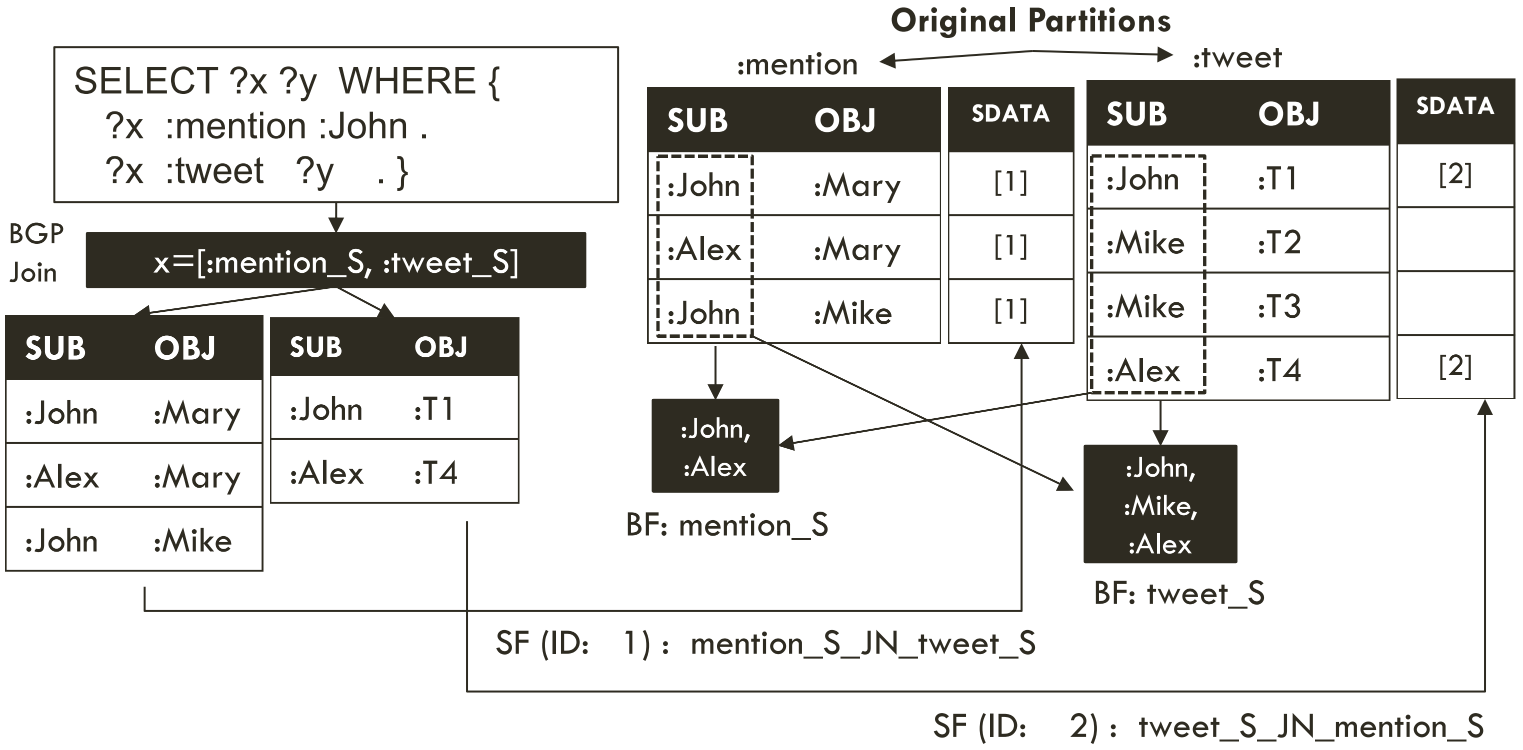
Semantic Vertical Partitioning (SemVP)

Semantic Data Layer

- **Semantic Filters**
  - Used for determining triples that can are read to answer a specific query join pattern
- **Statistics**
  - Maintains statistics about the semantic filters
- **Bloom Filters**
  - Used for computing the semantic filters

Property Reduction

Example



Partition Reduction

Example: Property Relatedness

- Properties are considered **related** if they appear in the **same query join pattern**
- SPARTI utilizes the **co-occurrence** to determine which semantic filters should be computed

```
SELECT ?X ?Y ?Z
WHERE
{
        ?X type       GraduateStudent .
        ?Z phdFrom    ?Y .
        ?X mscFrom    ?Y .
}
```

type = [mscFrom]
phdFrom = [mscFrom]
mscFrom = [type, phdFrom]

Evolution

- Evolution relates to when semantic filters are:
  - Created – To reduce partitions of newly observed join patterns
  - Deleted – To reduce disk space

- The cost of computing additional semantic filters can be inferred from the **query-workload** and the **SemVP statistics**

Cost Model

- Each **SemVP** partition maybe associated with **hundreds of semantic filters**

- A **cost-model** is needed in order to determine the **importance** of a semantic filter

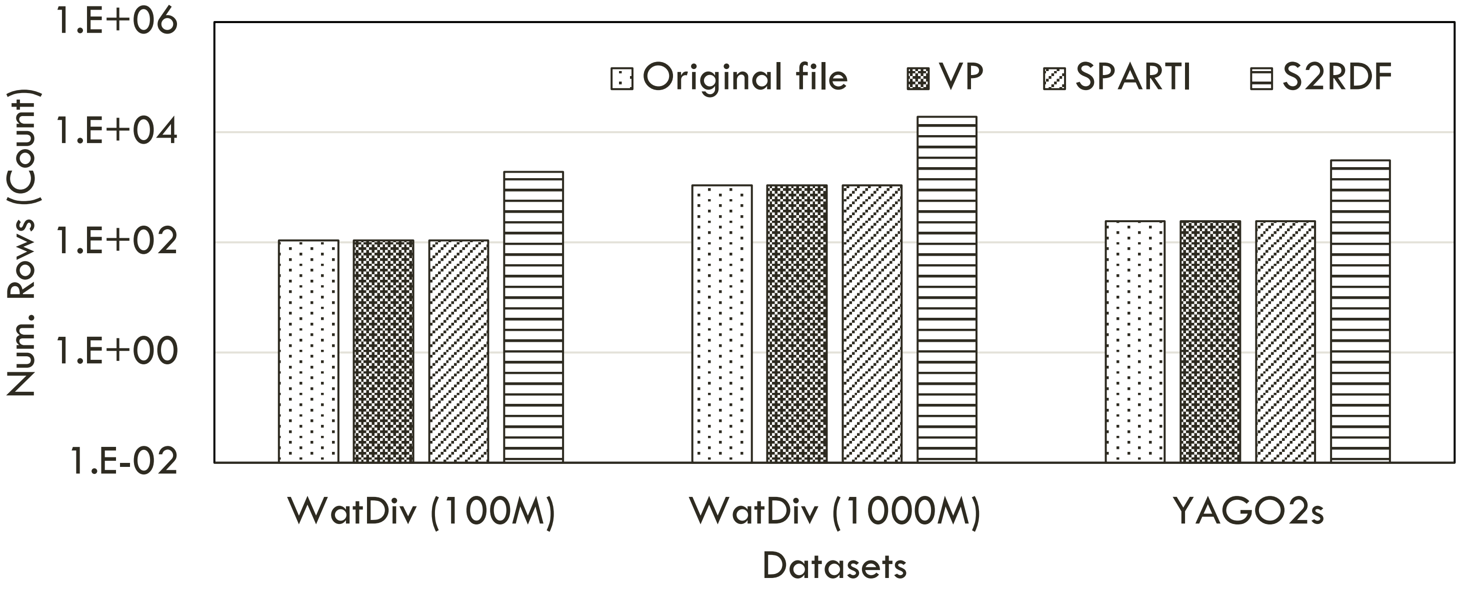
$$Utility = \alpha(S) + ß\,(R) + \infty\,(P)$$

- Where
  - S : Support (ie, frequency) of a join pattern within a query-workload
  - R : The partition size
  - P : Number of properties that the semantic filter includes
  - $\alpha + ß + \infty = 1$

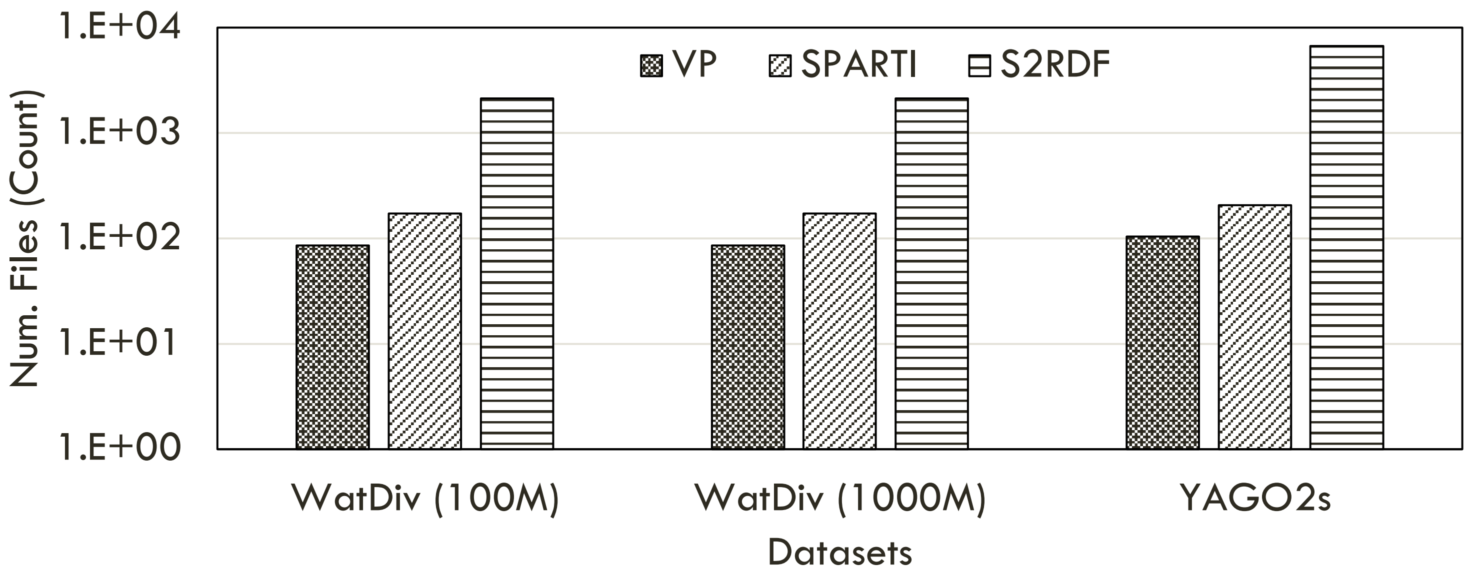
Experimental Setup

- **Computational Framework**
  - Apache Spark
- **Storage**
  - HDFS
- **Datasets**
  - WatDiv Benchmark + Stress Workload
    - 100 Million, 1 Billion
  - YAGO
    - 200 M
- **Systems**
  - S2RDF
  - SPARTI
  - Vertical Partitioning

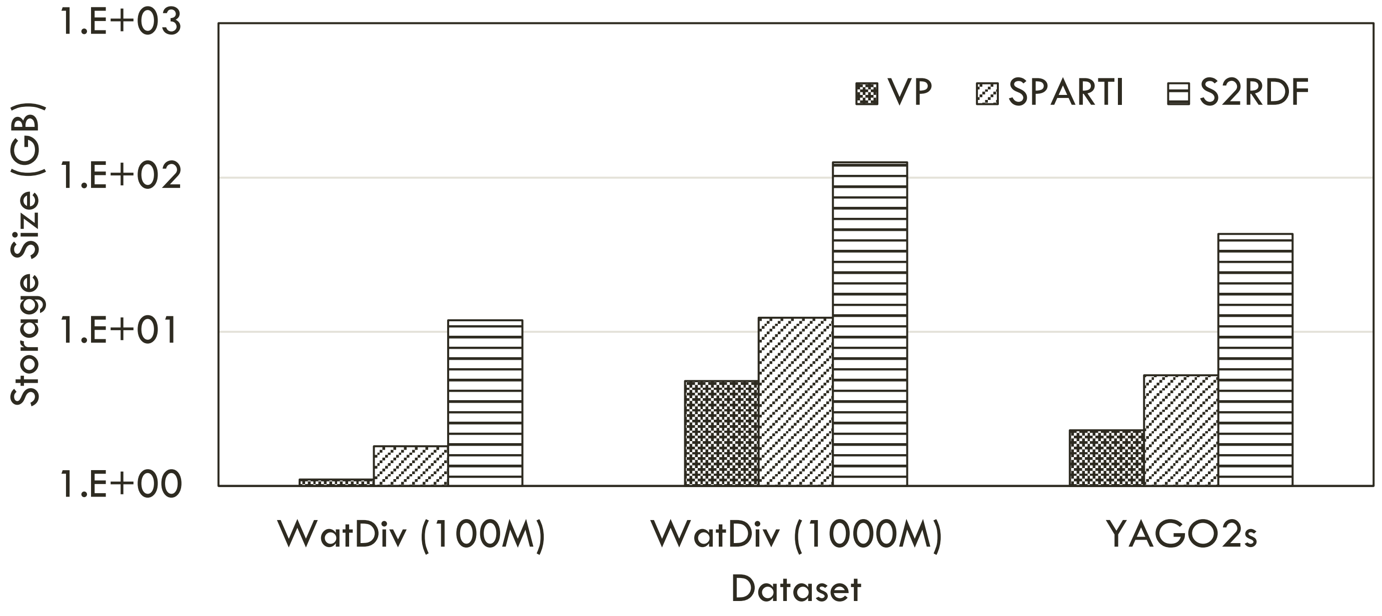
Results
Number of Rows



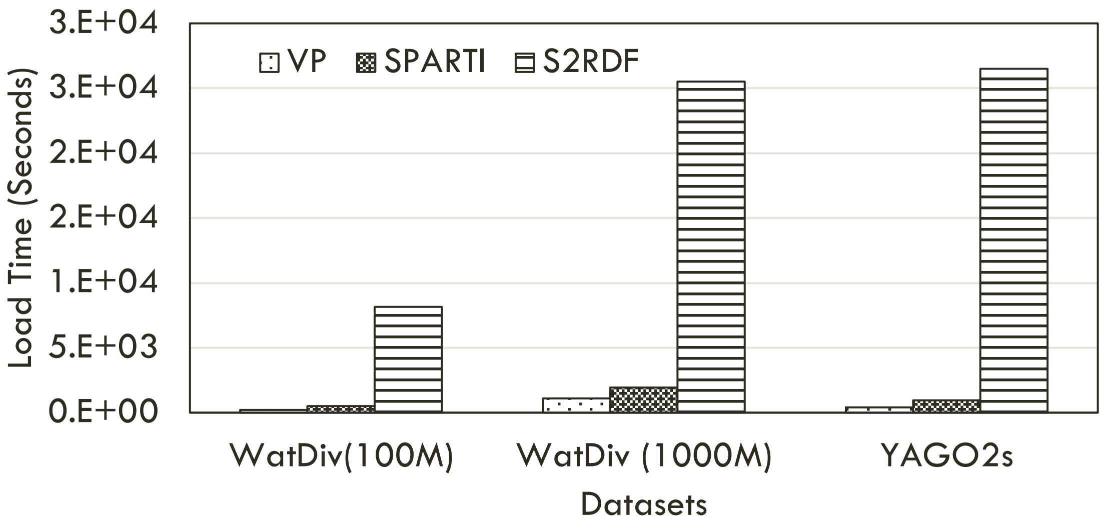
Results
Number of Files

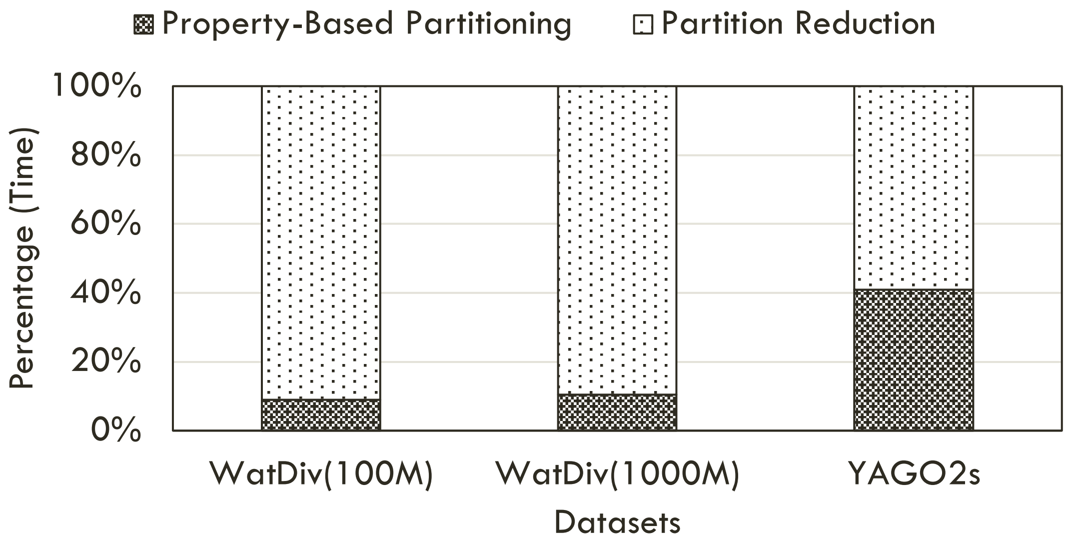


Results
HDFS Size

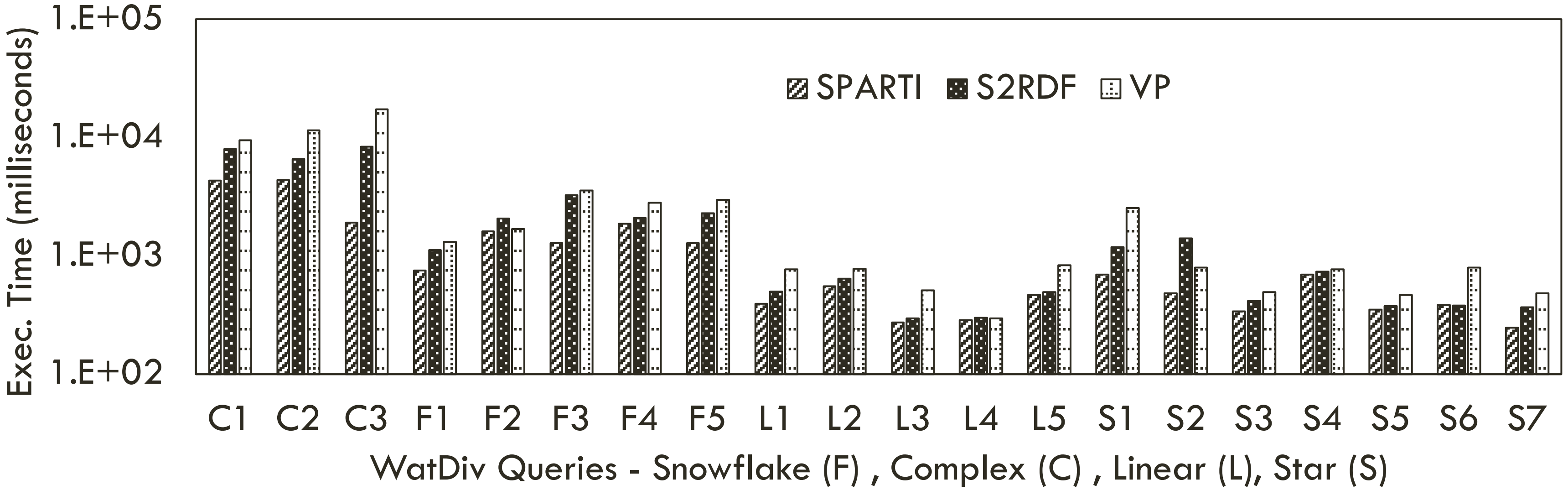


Results
Load Time



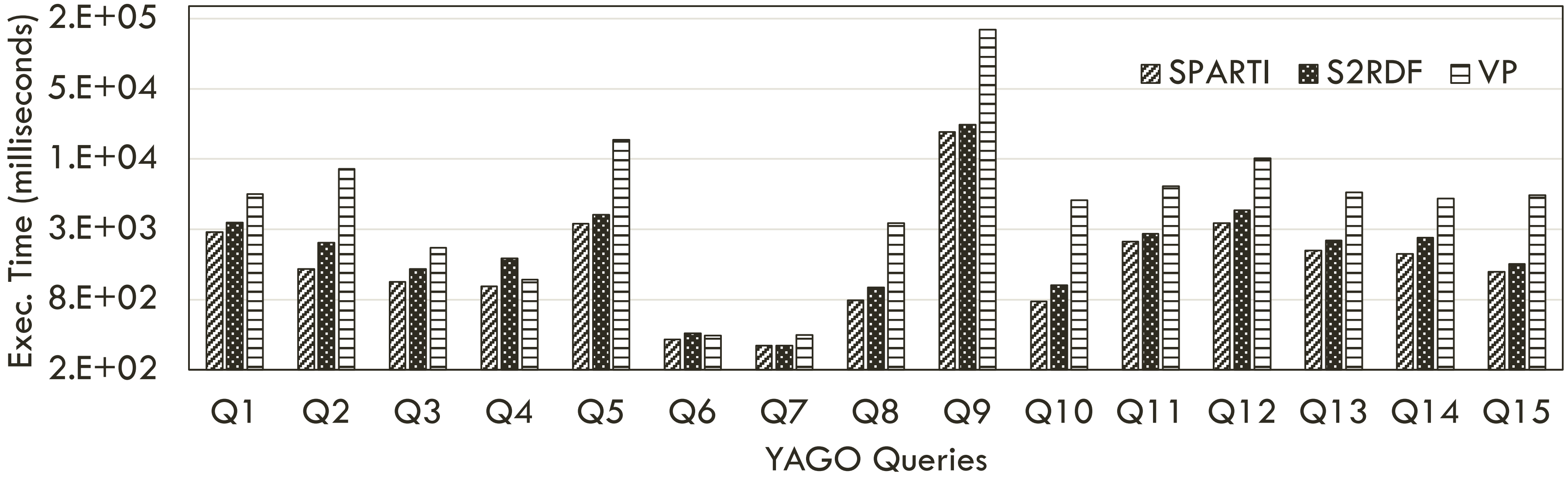


Results
Execution Time – WatDiv 1B



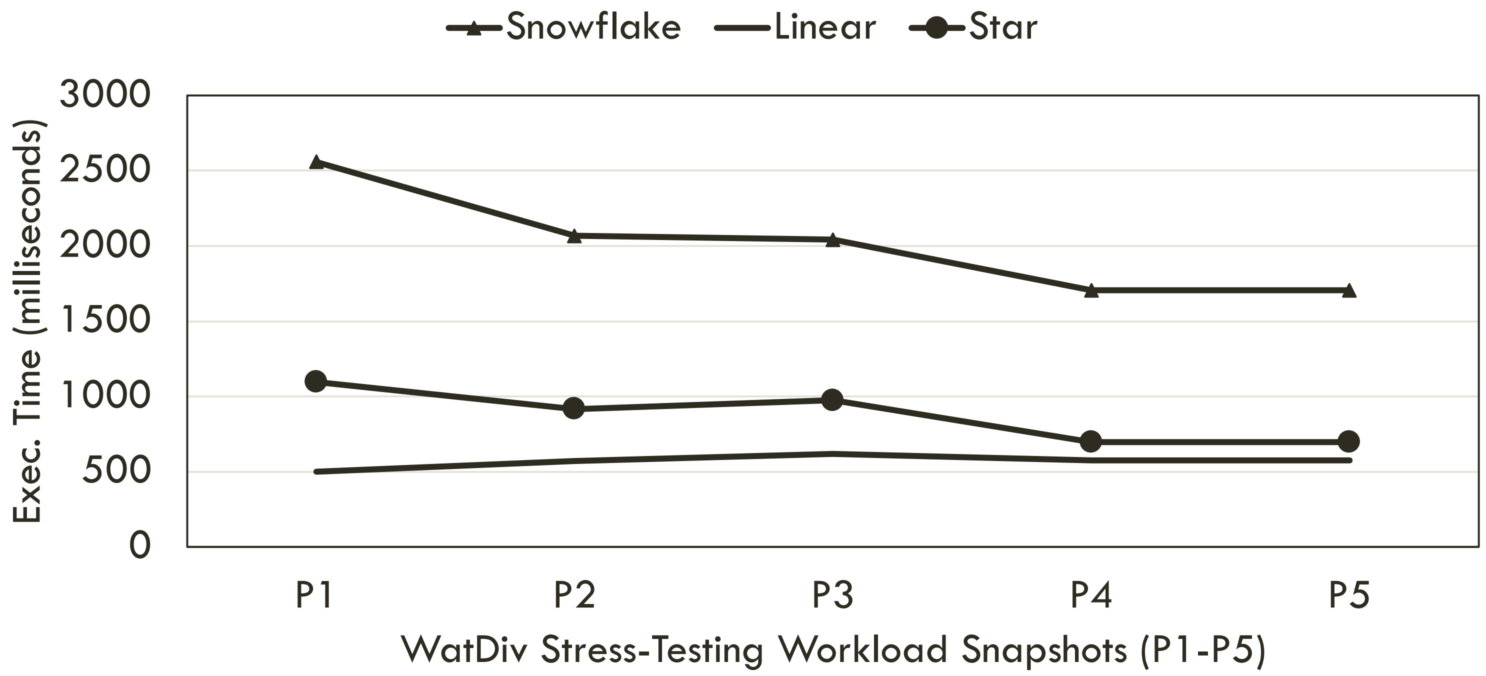
Results

Execution Time - YAGO



Results

Execution – WatDiv Stress Testing Workload



Conclusion

- We presented SPARTI, a **scalable** RDF data management system that utilizes a **relational scheme** and provides **row- level semantics** for RDF data

- The row level semantics, represented as **semantic filters**, provide SPARTI with a mechanism to read a reduced set of rows when answering specific **query join-patterns**

- The **cost-model** for managing semantic filters **prioritizes** the creation of the important semantic filters to compute.

- The **experimental study** that compares SPARTI with the state-of-the-art Spark-based RDF system demonstrates that SPARTI achieves **robust performance** over synthetic and real datasets

Questions ?

BACKUP SLIDES