HDT-MR

A Scalable Solution for RDF Compression with HDT and MapReduce

J.M. Giménez-García[1] J.D. Fernández[2] M.A. Martínez-Prieto[1]

[1]DataWeb Research, Department of Computer Science
University of Valladolid (Spain)

[2]Vienna University of Economics and Business (Austria)

ESWC, 2015

Outline

Overview

- RDF is commonly serialized in verbose formats.
  - Space overheads.
  - Consumption overheads.
- HDT: Binary serialization based on compact data structures.
  - Compressed space.
  - Efficient random access.
- HDT moves the scalability issues from consumer to publisher.
- HDT-MR uses MapReduce to deal with great volume HDT generation.

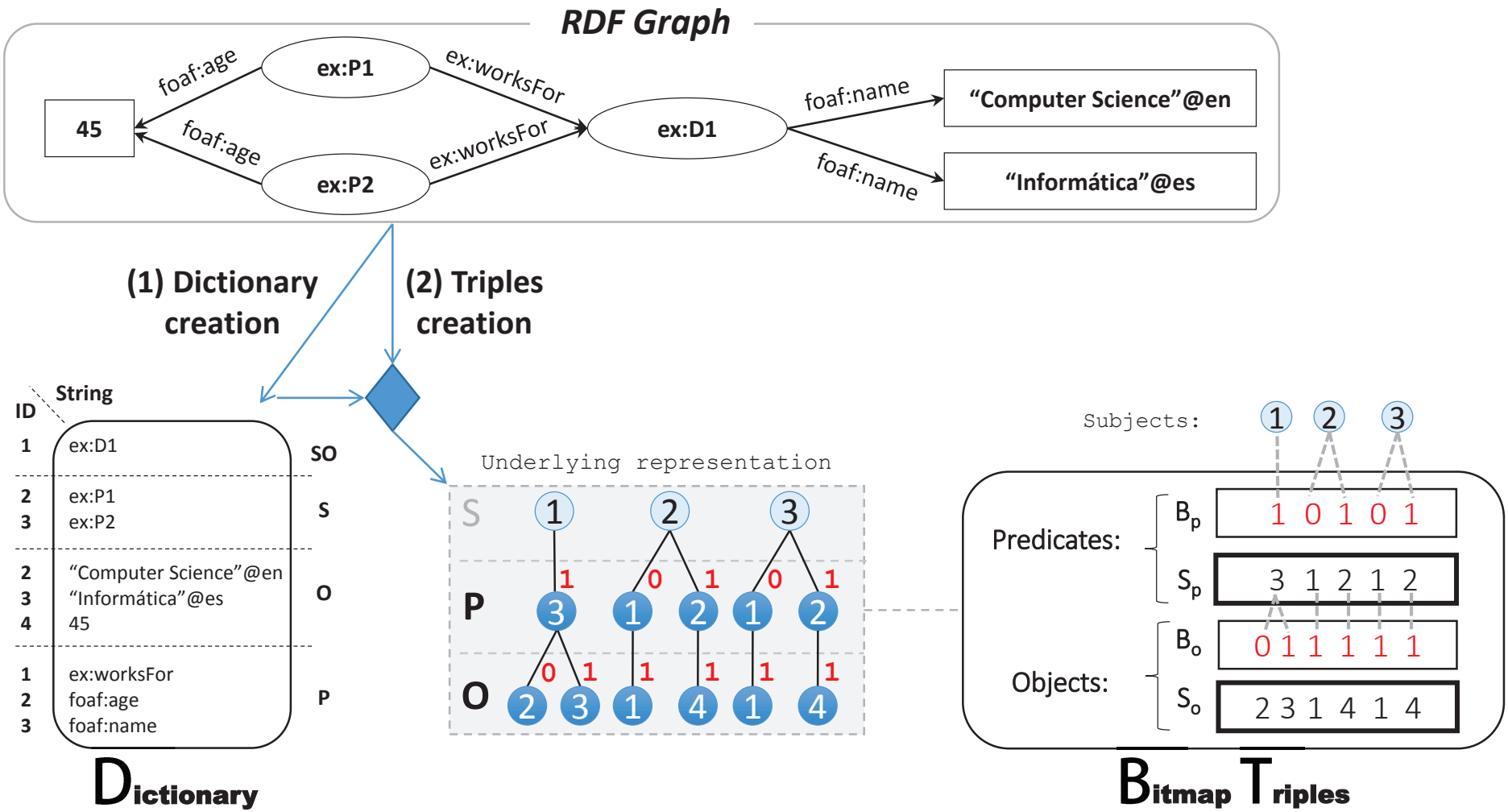
MapReduce

- Framework and programming model to process large amounts of data in a distributed way.
- Master/slave architecture.
- "Move the algorithm to the data"
- Exhaustive I/O operations and intensive in bandwidth usage.

Phases

map: $(k1, v1) \rightarrow list(k2, v2)$
reduce: $(k2, list(v2)) \rightarrow list(v2)$

HDT


RDF Graph
45
ex:P1
ex:P2
ex:D1
foaf:age
foaf:age
ex:worksFor
ex:worksFor
foaf:name
foaf:name
“Computer Science”@en
“Informática”@es
(1) Dictionary creation
(2) Triples creation
ID
String
1 ex:D1
2 ex:P1
3 ex:P2
2 “Computer Science”@en
3 “Informática”@es
4 45
1 ex:worksFor
2 foaf:age
3 foaf:name
SO
S
O
P
Dictionary
Underlying representation
S
P
O
Subjects:
Predicates:
Objects:
B_p
S_p
B_o
S_o
1 0 1 0 1
3 1 2 1 2
0 1 1 1 1 1
2 3 1 4 1 4
Bitmap Triples

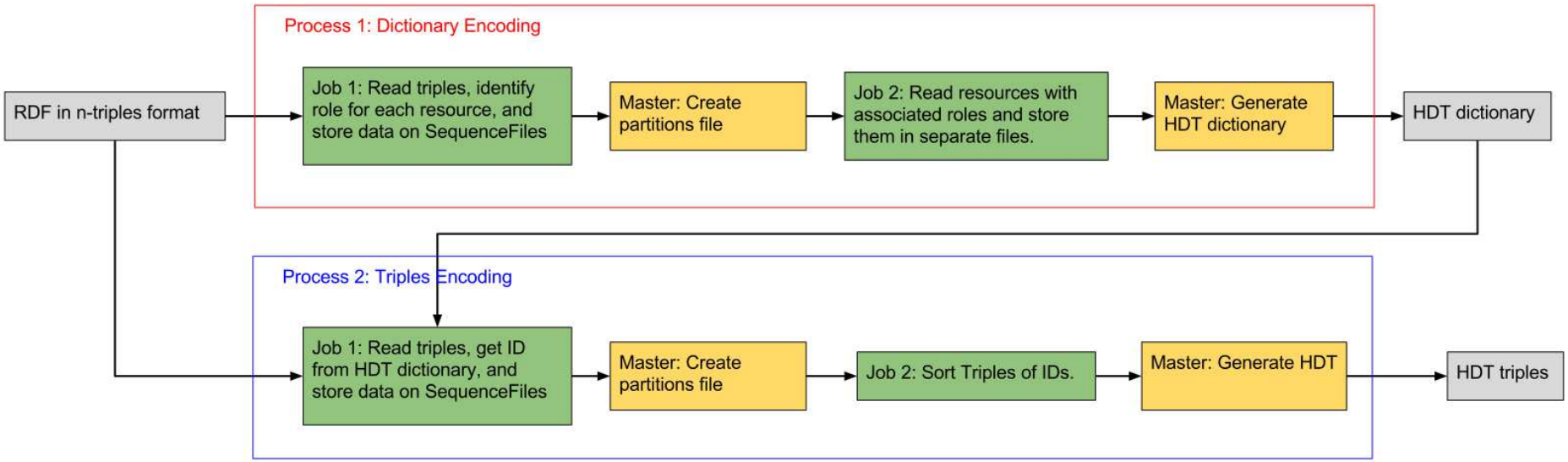
HDT

Serialization process

- Classifying RDF terms.
  - Triple-by-triple parsing.
  - Three hash tables: S, P, O.
  - Triples are encoded by temporary ID
  - SO hash table is built after parsing.
- Building HDT Dictionary.
  - Each section is sorted lexicographically.
  - Relation between definitive IDs and temporary IDs are stored.
- Building HDT Triples.
  - Triples have their temporary IDs replaced by definitive IDs.
  - ID-triples are sorted by subject, predicate and object ID
  - Bitmap Triples are obtained.

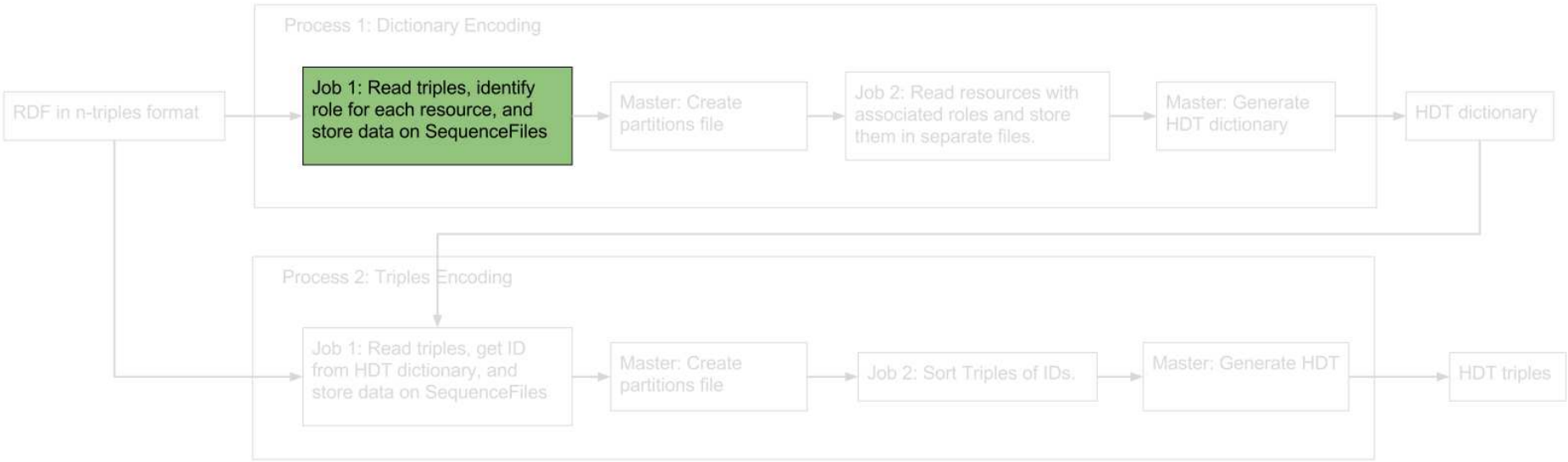
HDT-MR

Global Perspective



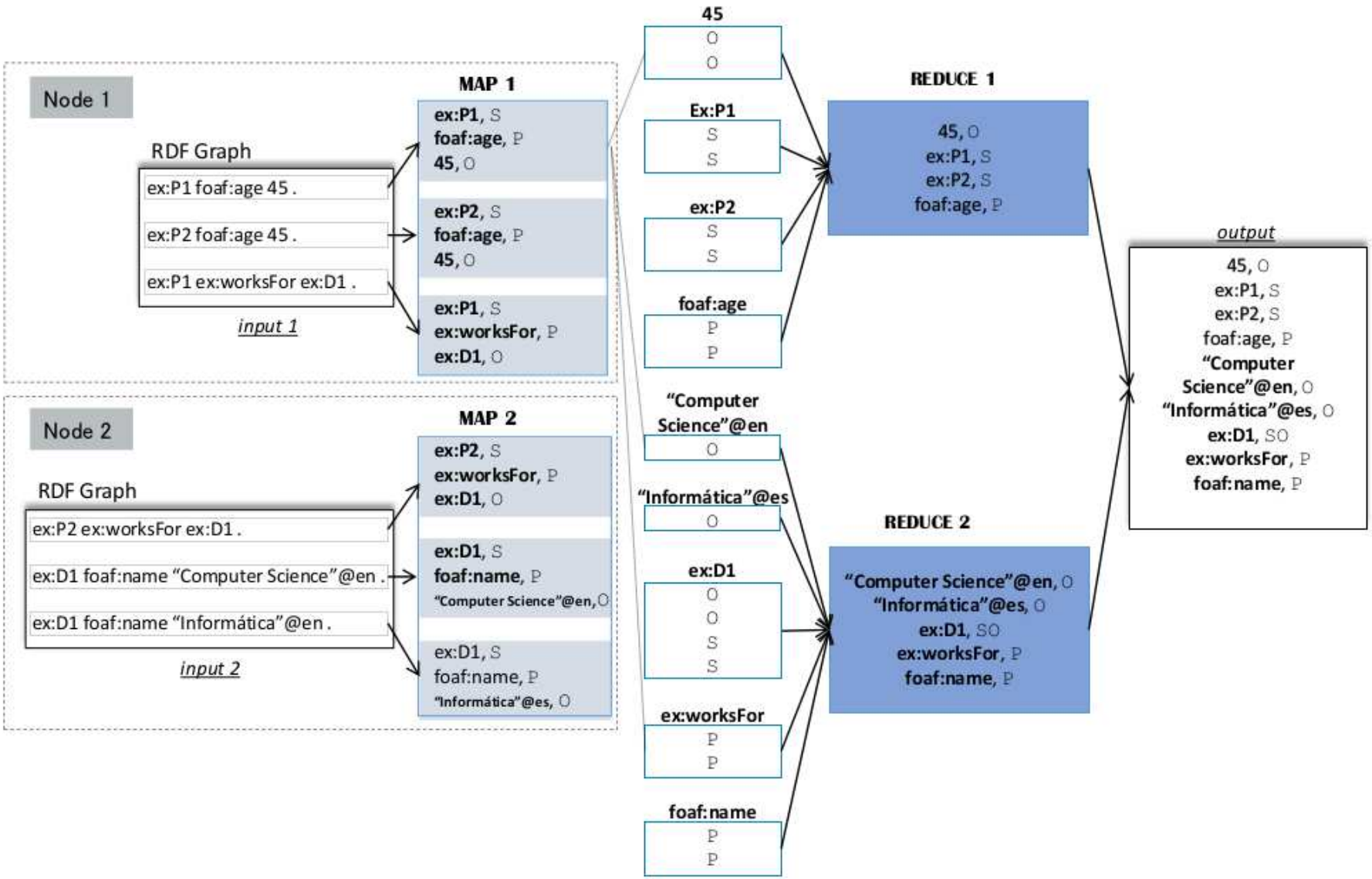
HDT-MR

Dictionary Encoding - Job 1



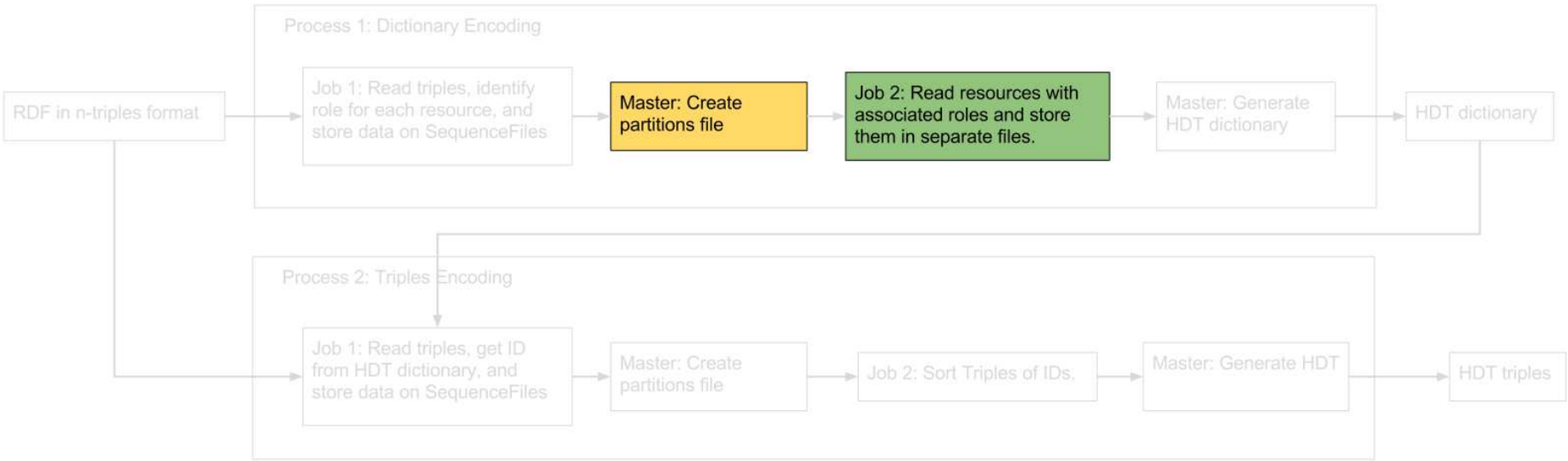
HDT-MR

Dictionary Encoding - Job 1



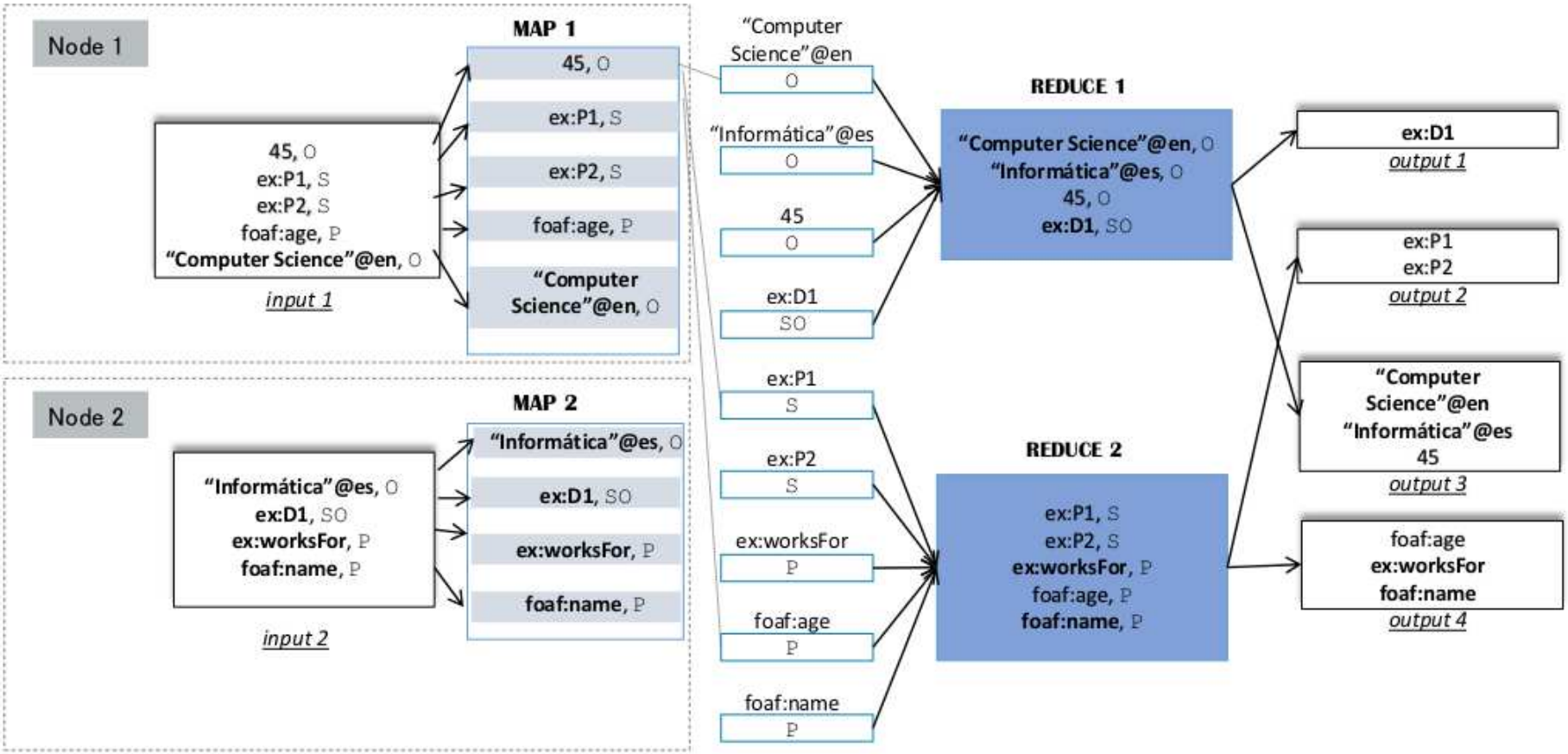
HDT-MR

Dictionary Encoding - Job 2



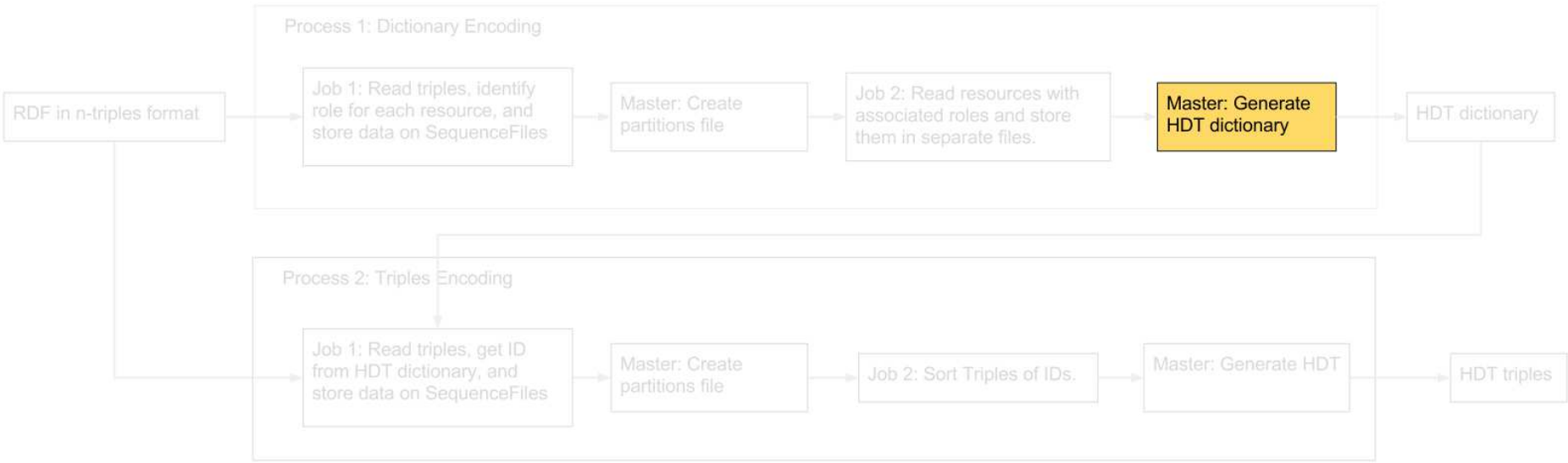
HDT-MR

Dictionary Encoding - Job 2



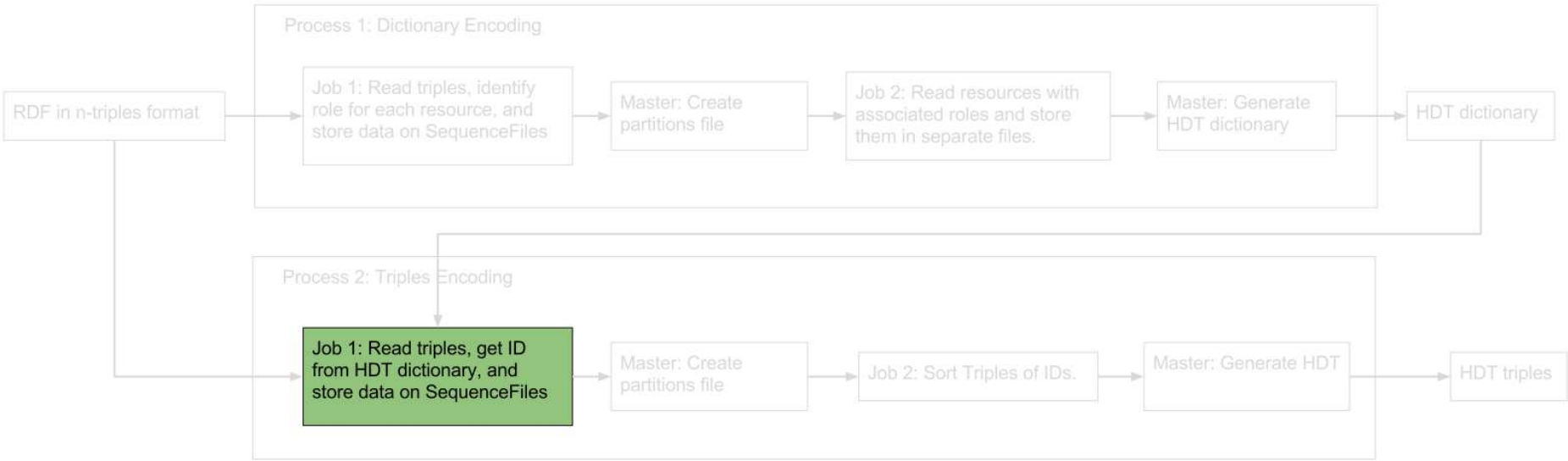
HDT-MR

Dictionary Encoding - Generate HDT Dictionary



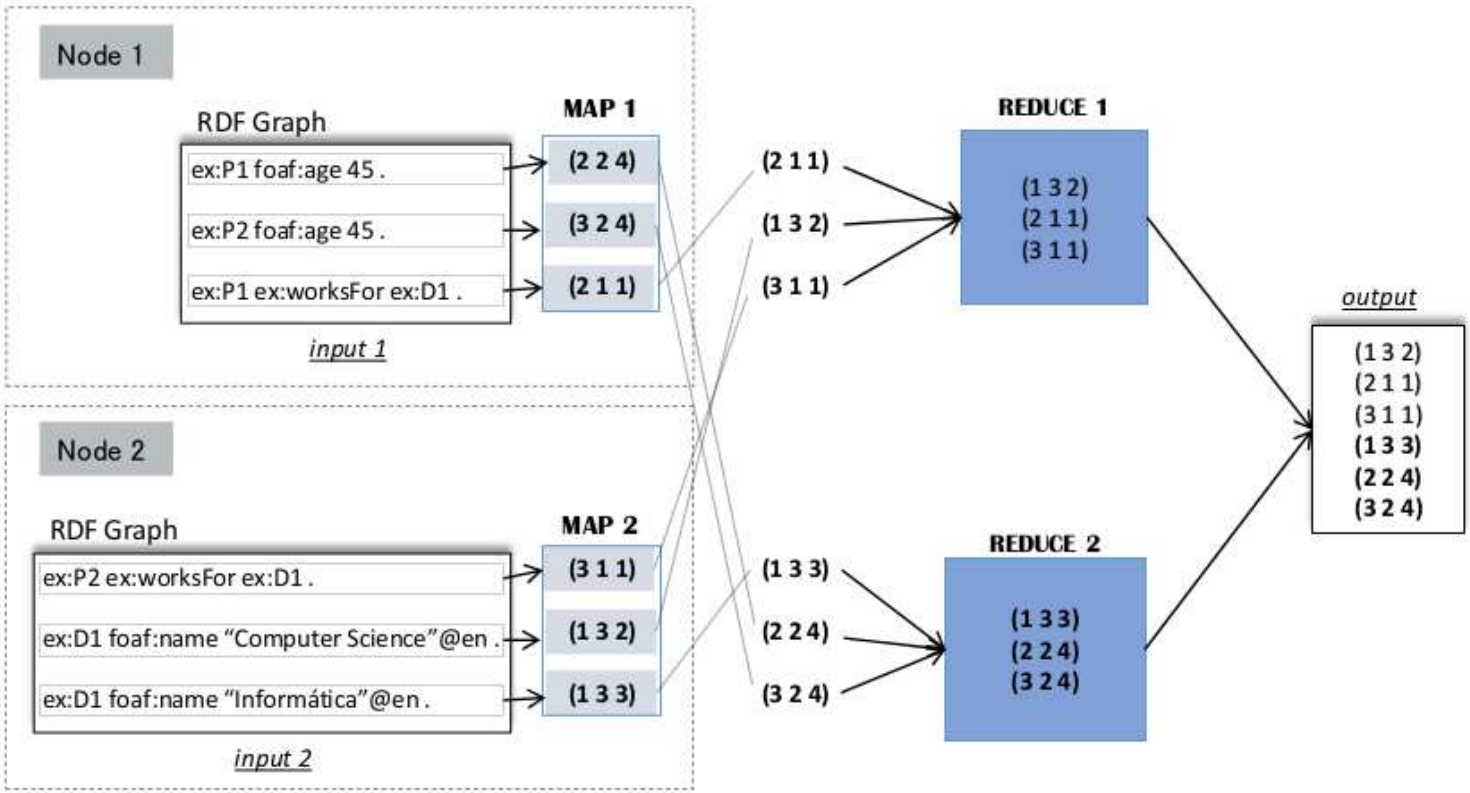
HDT-MR

Triples Encoding - Job 1



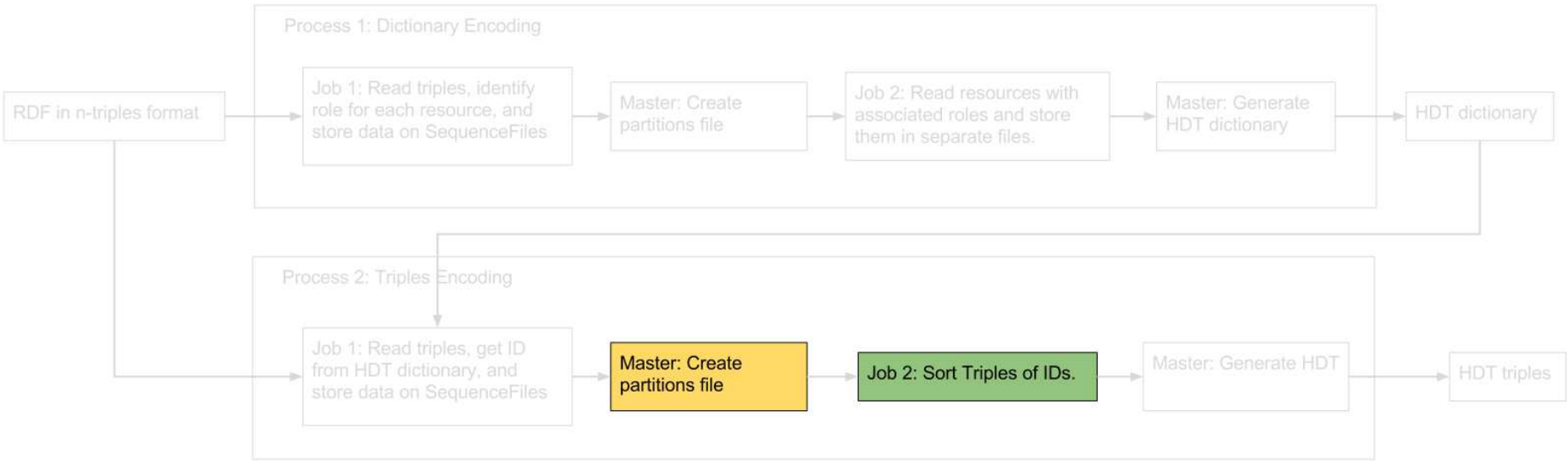
HDT-MR

Triples Encoding - Job 1



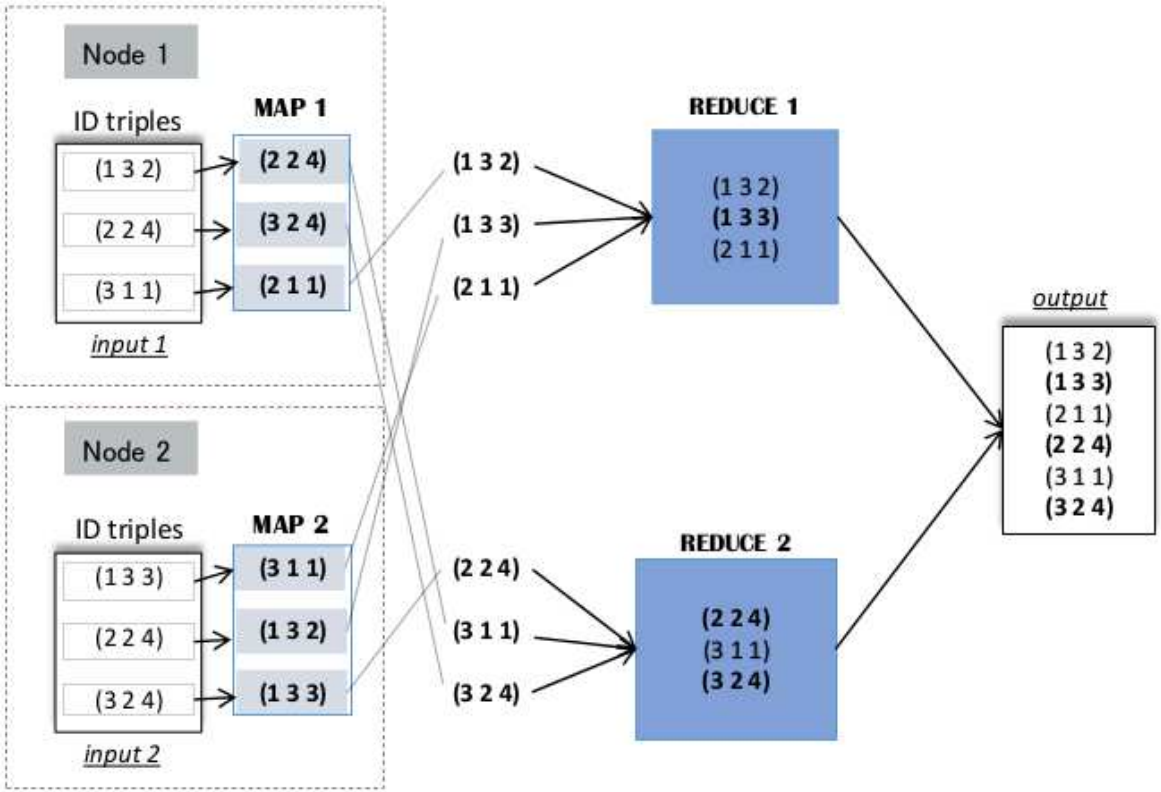
HDT-MR

Triples Encoding - Job 2



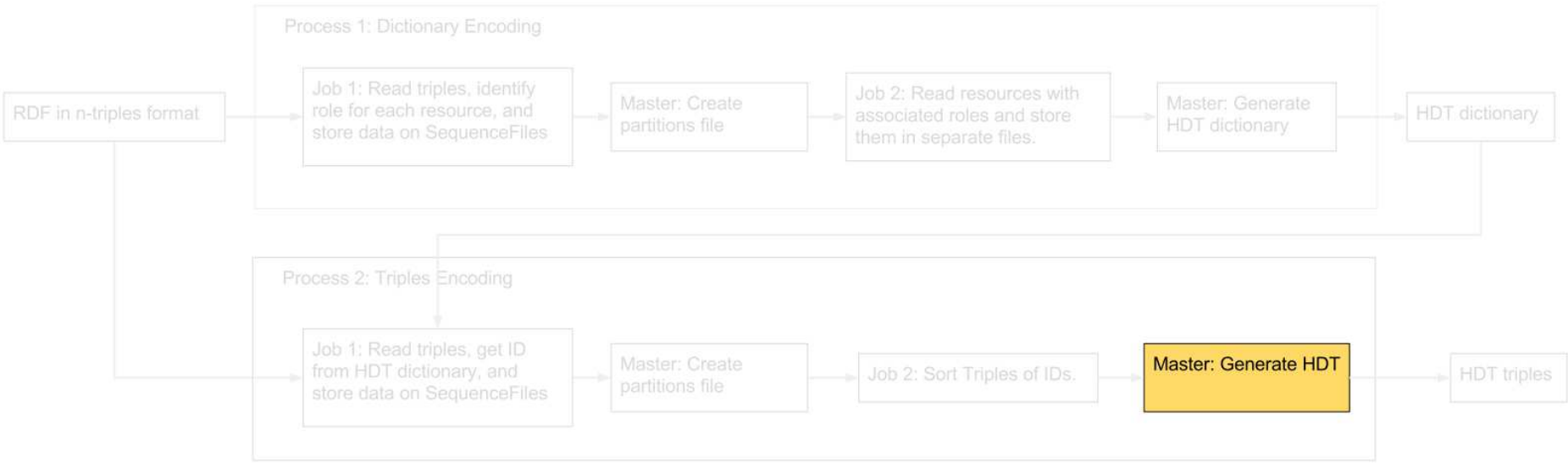
HDT-MR

Triples Encoding - Job 2



HDT-MR

Triples Encoding - Generate Partitions File



Experimental Evaluation

Experiments Design

Machine	Configuration
1 x Single Node	Intel Xeon E5-2650v2 @ 2.60GHz (32 cores), 128GB RAM. Debian 7.8
1 x Master	Intel Xeon X5675 @ 3.07 GHz (4 cores), 48GB RAM. Ubuntu 12.04.2
10 x Slaves	Intel Xeon X5675 @ 3.07 GHz (4 cores), 8GB RAM. Debian 7.7

- Hadoop 1.2.1

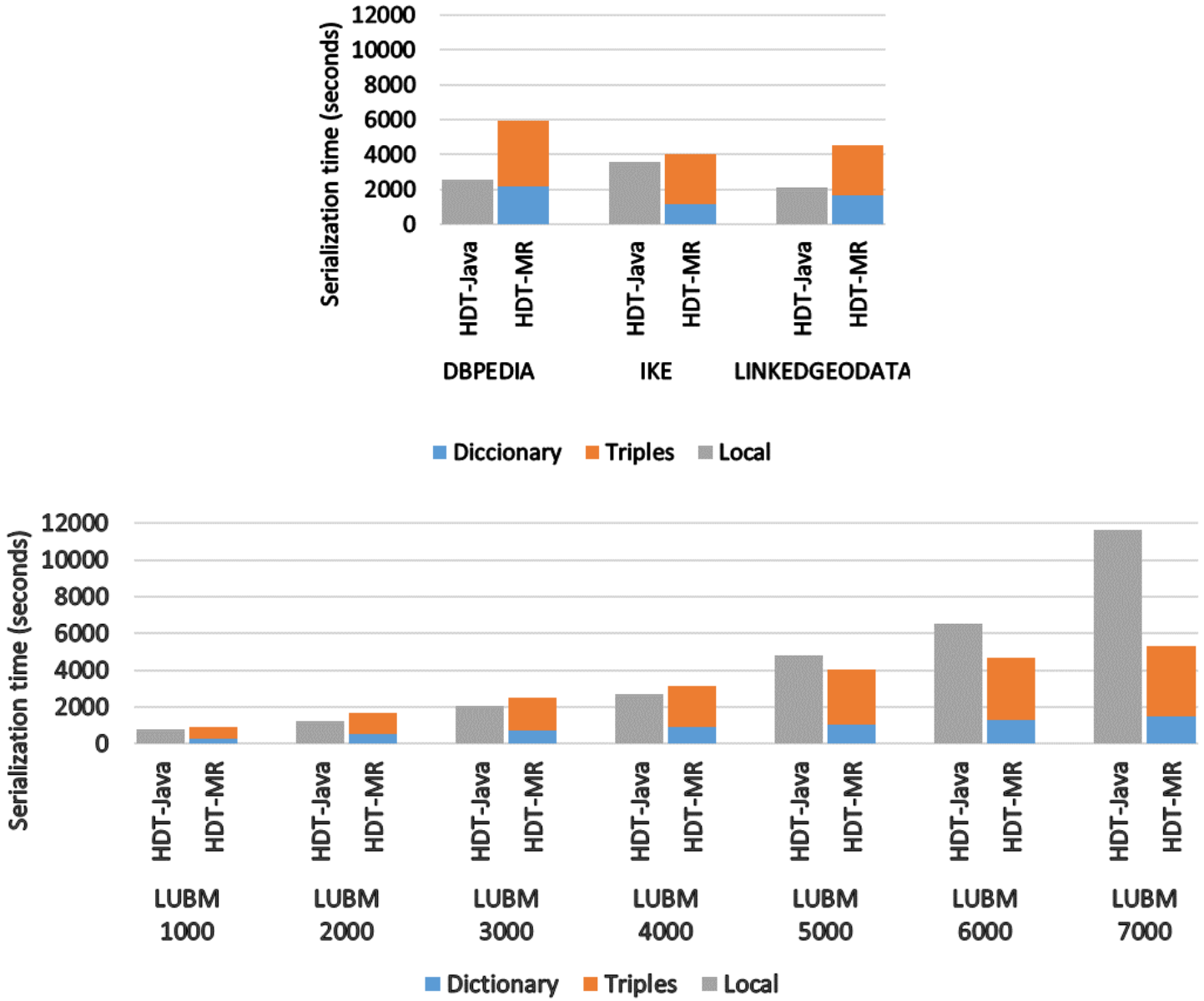
Experimental Evaluation

Experiments Design

Dataset	Triples	|SO|	|S|	|O|	|P|	Size (GB) NT	NT+lzo	HDT	HDT+gz
LinkedGeoData	0.27BN	41.5M	10.4M	80.3M	18.3K	38.5	4.4	6.4	1.9
DBPedia	0.43BN	22.0M	2.8M	86.9M	58.3K	61.6	8.6	6.4	2.7
Ike	0.51BN	114.5M	0	145.1K	10	100.3	4.9	4.8	0.6
Mashup	1.22BN	178.0M	13.2M	167.2M	76.6K	200.3	18.0	17.1	4.6
LUBM-1000	0.13BN	5.0M	16.7M	11.2M	18	18.0	1.3	0.7	0.2
...	...	...	...	...	...	...	...	...	...
LUBM-5000	0.67BN	24.9M	83.7M	55.8M	18	90.9	6.6	3.9	1.3
...	...	...	...	...	...	...	...	...	...
LUBM-8000	1.07BN	39.8M	134.0M	89.3M	18	145.5	10.6	6.3	2.2
...	...	...	...	...	...	...	...	...	...
LUBM-40000	5.32BN	198.4M	666.7M	444.5M	18	730.9	52.9	33.2	10.4

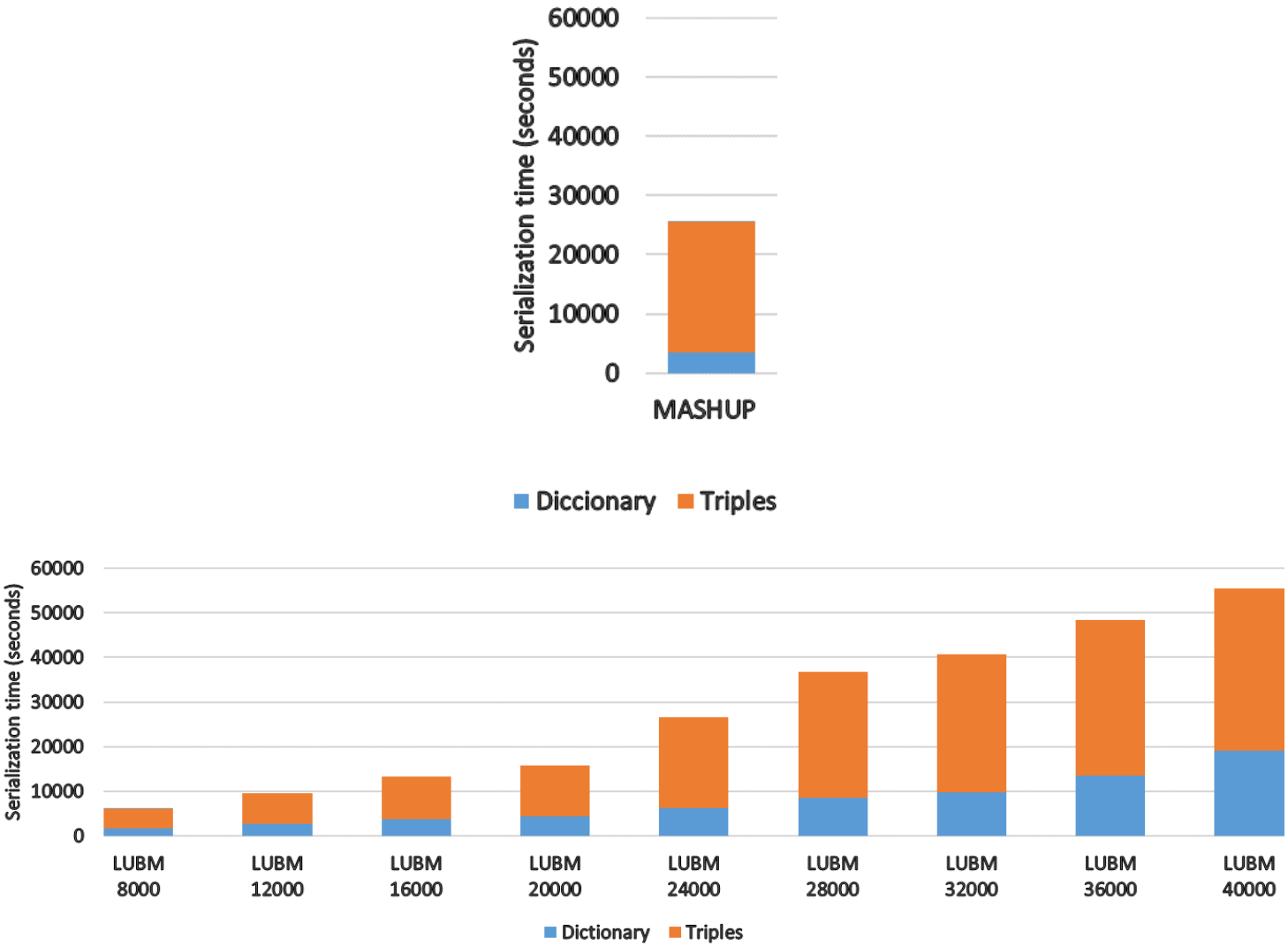
Experimental Evaluation

Results



Experimental Evaluation

Results



Conclusions and Future Work

- HDT-MR ...
  - tackles scalability issues arising in HDT construction at very large scale.
  - lightens the previous heavy memory-consumption burden by moving the construction task to the MapReduce paradigm.
  - is evaluated in huge real-world and benchmark RDF datasets scaling up to more than 5 billion triples
- Future work
  - Exploit HDT-MR achievements and fostering the development of novel applications working at very large scale.
  - Combine HDT and MapReduce foundations to work together on other Big Semantic Data tasks, such as querying and reasoning.
- Do you want to know more?
  - `http://www.rdfhdt.org`

HDT-MR

A Scalable Solution for RDF Compression with HDT and MapReduce

J.M. Giménez-García[1] J.D. Fernández[2] M.A. Martínez-Prieto[1]

[1]DataWeb Research, Department of Computer Science
University of Valladolid (Spain)

[2]Vienna University of Economics and Business (Austria)

ESWC, 2015