SEMANTiCS
Amsterdam 2021

3rd International Workshop On Semantics And The Web For Transport (Sem4Tra), Semantics 2021

Semantic Conversion of Transport Data Adopting Declarative Mappings: An Evaluation of Performance and Scalability

Mario Scrocca, Marco Comerio, Alessio Carenini and Irene Celino

name.surname@cefriel.com

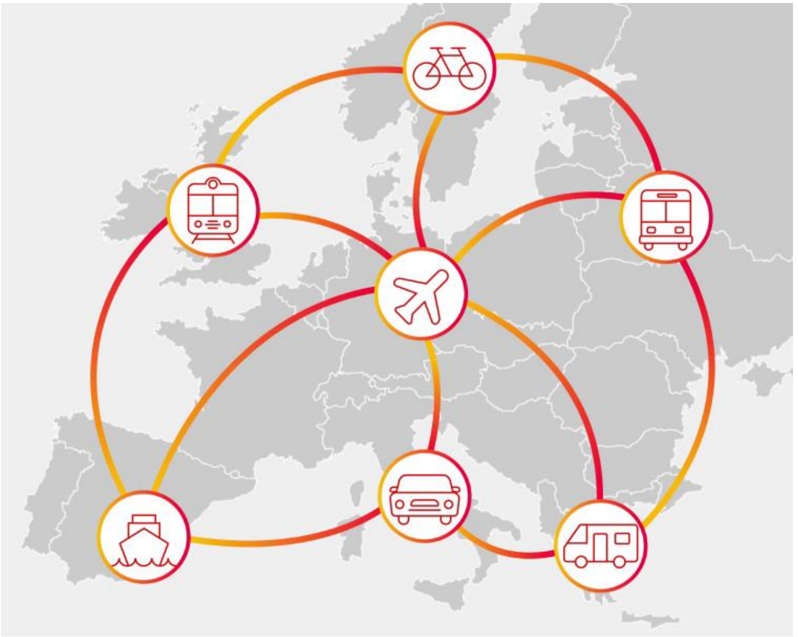
Cefriel
POLITECNICO DI MILANO

sprint

INCREASING INTEROPERABILITY IN THE TRANSPORTATION DOMAIN

Shift2Rail IP4[1] is aiming at improving the European transportation landscape

- Increase the multimodal usage and the total number of passengers
- Increase the usage of cross-border train services
- Improve the quality of services facilitating the travel planning for users
- Reduce costs



All those objectives require an improved collaboration among Transport Operators

- There is no single «interoperability problem» and therefore **no single «interoperability solution»**
- Need for a **collection of specialized tools** that can be combined and deployed autonomously in heterogeneous environments
- Industrial adoption requires **performant** and **scalable** solutions

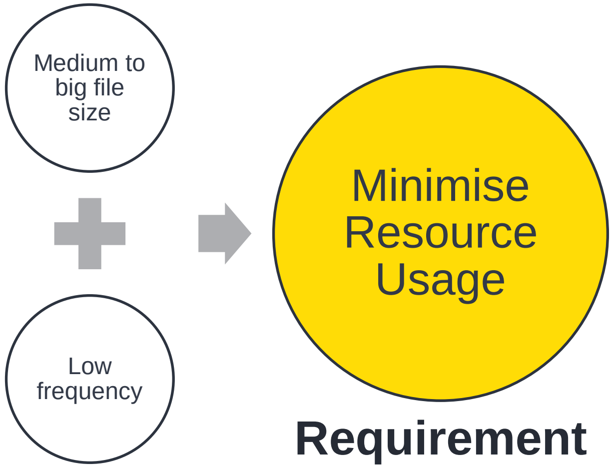
[1] https://shift2rail.org/research-development/ip4/

Goal: Achieve interoperability letting stakeholders keep using their current legacy systems

Many different formats, standards and specifications

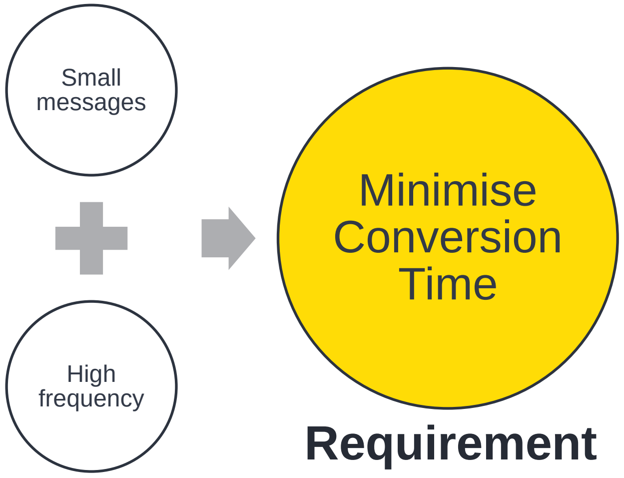
Two main information exchange patterns, two conversion patterns

Dataset-based (Batch conversion)



Requirement

Message-based (Service mediation)



Requirement

THE IP4 INTEROPERABILITY FRAMEWORK IN SPRINT (http://sprint-transport.eu/)

Transport domain knowledge is represented using **Ontologies**

Mappings define rules from different formats/specifications/standard onto such ontologies

Mappings and ontologies are leveraged by **Converters** to obtain data interoperability

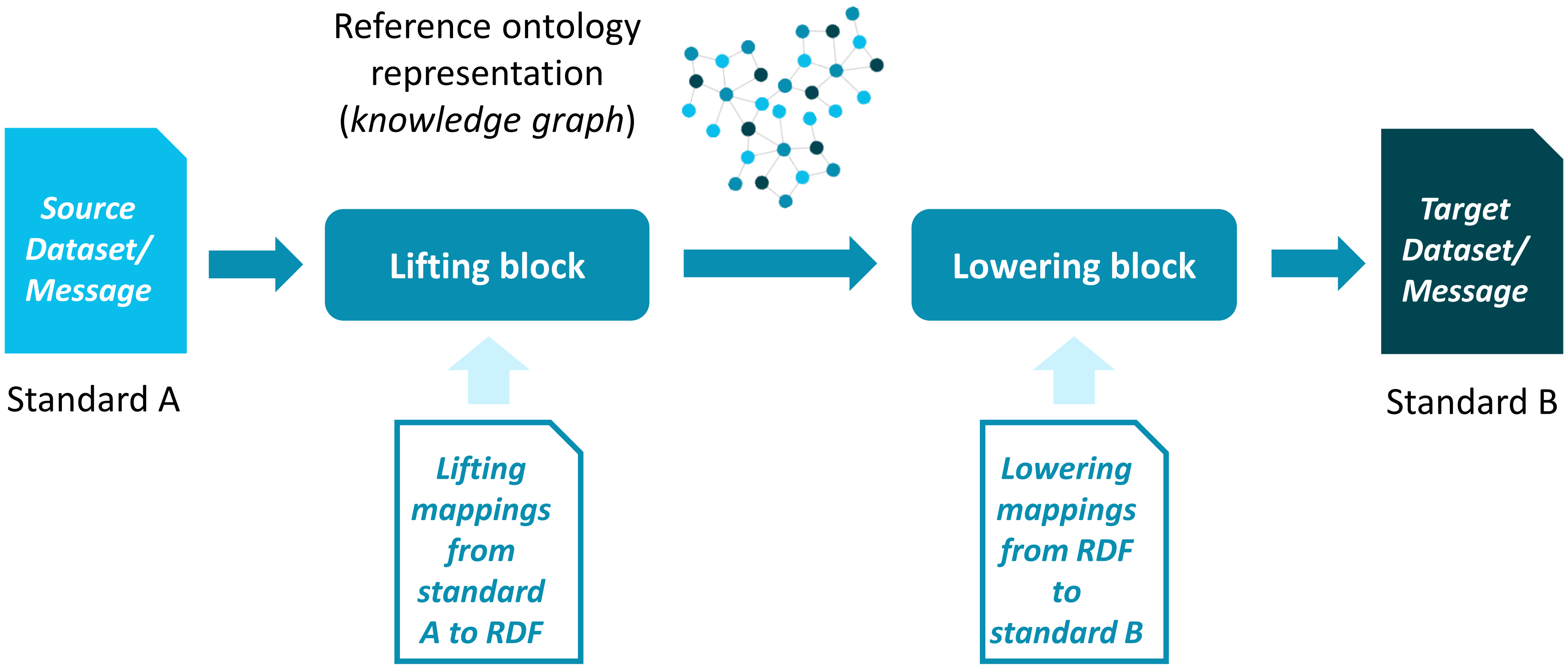
Ontologies, Mappings, Converters, API descriptions and Datasets are shared using a **Catalogue** which enforces Governance

Video: SPRINT at a Glance

Declarative mappings define the conversion rules using Semantic Web technologies

- **Lifting** rules: we «extract knowledge» from the input message according to a reference ontology
- **Lowering** rules: we use such knowledge to build the output message

We implemented and tested a **semantic converter** relying on declarative mappings based on the *Chimera* framework.



CONVERSION ALTERNATIVES: ST4RT JAVA ANNOTATIONS

Annotation-based conversion defined in the ST4RT project exploits **Java annotations** to define mappings for Java classes

Lifting:

- marshall from source data format to Java instances
- unmarshall from Java instances to RDF

Lowering

- unmarshall from RDF to Java instances
- marshall from Java instances to target data format

This approach is **useful for web-services where an interface descriptor is used** (Java classes can be automatically generated and annotated).

Drawbacks of the ST4RT approach:

- an object-oriented representation of the source/target data format is required,
- performance and scalability issues arise for conversion procedures with complex annotations and/or handling large files.

```java
@Namespaces({"tm", "https://w3id.org/transmodel/terms#"})

@RdfsClass("tm:Agency")
public class Agency {

    @RdfProperty(propertyName = "tm:id")
    private String id;

    @RdfProperty(propertyName = "tm:name")
    private String name;

}
```

Carenini, A., et al.: ST4RT – Semantic Transformations for Rail Transportation. In: 7th Transport Research Arena (TRA 2018). Zenodo (Apr 2018). https://doi.org/10.5281/zenodo.1440984

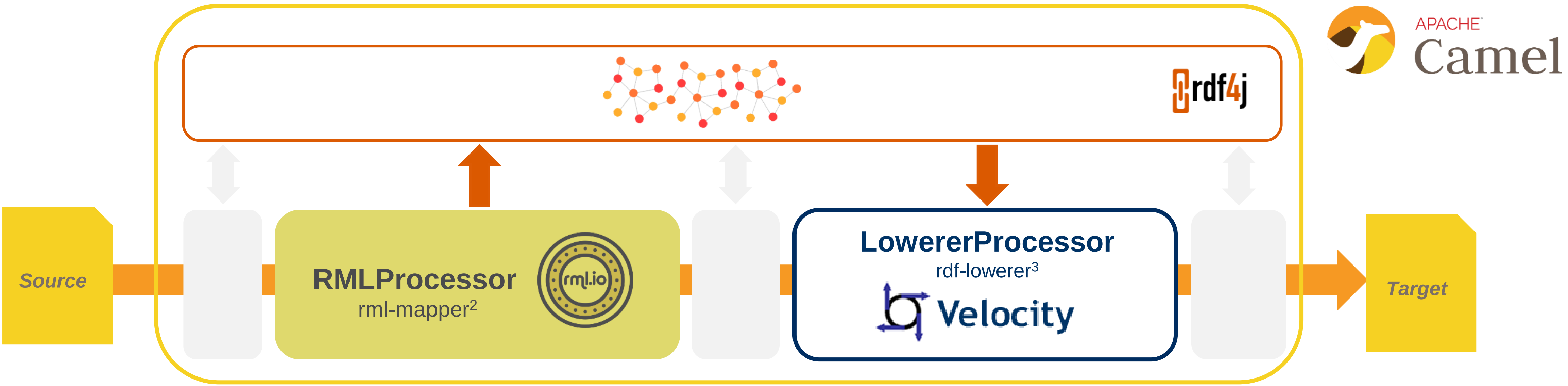
CHIMERA: SEMANTIC DATA TRANSFORMATION PIPELINES

Chimera[1] is an open-source framework built on top of *Apache Camel* and implementing specific components to enable conversion pipelines based on Semantic Web technologies

- **Modular solution** to minimise the effort required to customize a pipeline.
- **Fully declarative** approach:

  Lifting: RDF Mapping Language (RML)

  Lowering: Apache Velocity templates with embedded SPARQL queries



[1]**Chimera**: https://github.com/cefriel/chimera
[2] fork based on the RMLMapper https://github.com/cefriel/rmlmapper-cefriel
[3]*rdf-lowerer* custom implementation https://github.com/cefriel/rdf-lowerer

RML LIFTING

- This block accepts mappings defined through the **RML mapping language** (mapping rules from different data formats to an RDF serialization)

- RML extends R2RML (SQL) allowing also mappings from **heterogeneous data sources** (CSV, XML, JSON) through the definition of iterators.

- Built-in support:
  - for data enrichment between multiple input data sources (also with different data formats)
  - to apply custom functions in processing the input data

```
<AuthorityMapping>
  a rr:TriplesMap;

  rml:logicalSource [
    rml:source "agency.csv";
    rml:referenceFormulation ql:CSV;
  ];

  rr:subjectMap [
    rr:template "agencies/{agency_id}";
    rr:class tm:Authority
  ];

  rr:predicateObjectMap [
    rr:predicate tm:name;
    rr:objectMap [
      rml:reference "agency_name"
    ]
  ];

  rr:predicateObjectMap [
    rr:predicate tm:id;
    rr:objectMap [
      rml:reference "agency_id"
    ]
  ].
```

APACHE VELOCITY TEMPLATES WITH EMBEDDED SPARQL QUERIES

- A template-based approach to guarantee maximum flexibility on the output format
- It can evaluate a generic Apache Velocity template (https://velocity.apache.org) replacing at runtime variables with actual values
- It offers specific functions to query an RDF graph and build a structured document
- SPARQL queries are defined inside the template, they are executed before the evaluation and bound to template variables
- Such variables are then used to access the queries' results while evaluating the specified logic to fill the template

```
#set ( $query = "
    SELECT ?id ?name
    WHERE {
        ?a <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
           <https://w3id.org/transmodel/terms#Authority> .
        ?a <https://w3id.org/transmodel/terms#name> ?name .
        ?a <https://w3id.org/transmodel/terms#id> ?id . })
#set ( $authorities = $reader.executeQuery($query) )
<?xml version="1.0" encoding="iso-8859-1"?>
<PublicationDelivery version="1.0"
xsi:schemaLocation="http://www.netex.org.uk/netex../../../xsd/
xmlns="http://www.netex.org.uk/netex" xmlns:xsi="http://www.w3
    <dataObjects>
      <ResourceFrame>
        <organisations>
           #foreach($authority in $authorities)
           <Authority id="$authority.id">
             <Name>$authority.name</Name>
           </Authority>
           #end
        </organisations>
      </ResourceFrame>
    </dataObjects>
</PublicationDelivery>
```

PERFORMANCE AND SCALABILITY TESTING

We performed performance and scalability testing considering the requirements of the two use cases.

Batch Conversion

- *Conversion*: GTFS to Linked GTFS and back to GTFS CSV
- *Datasets and Mappings:* **GTFS-Madrid Benchmark**[1] datasets (formats CSV, JSON, XML) and mappings.
- *Performance:* Conversion time
- *Scalability*: tested w.r.t the **size of the input dataset**: scale 1,5,10,50,100

Service Mediation

- *Conversion*: Response message from **HaCon VBB journey planning endpoint to TRIAS format**
- *Datasets and Mappings*: VBB TripList input message (representing travel
- solutions for a requested itinerary), custom lifting and lowering mappings adopting an extension of the IT2Rail ontology.
- *Performance:* Conversion time
- *Scalability*: JMeter used to test the scalability with an **increasing number of concurrent requests** (number of threads: 10, 50, 100, 150, 500, 1000, 2500, 5000; ramp-up period: 1 second; loop count: 1)

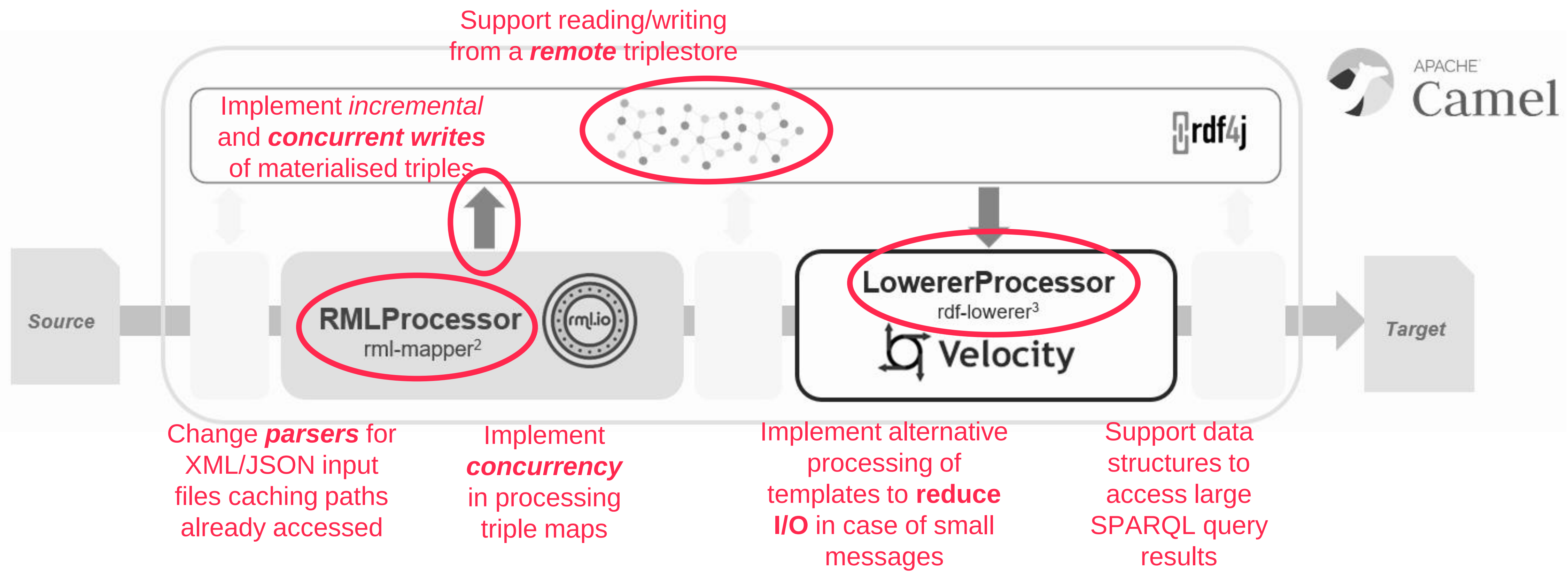
Configuration: Docker Containers on a machine running CentOS Linux 7, with Intel Xeon 8-core CPU and 64 GB Memory (24 GB – `memory-limit` given to the container running the Chimera Pipeline, 24h timeout for the conversion, GraphDB 9.0.0 Free by Ontotext used for tests with a remote repository)

[1] Chaves-Fraga, D., et al.: GTFS-Madrid-Bench: A benchmark for virtual knowledge graph access in the transport domain. Journal of Web Semantics 65, 100596 (2020). https://doi.org/10.1016/j.websem.2020.100596

TESTING PERFORMANCE AND SCALABILITY

We performed the testing activities considering two releases of **Chimera.** The **final release (f-rel)** introduces several optimisations based on the preliminary results obtained for the **core release (c-rel).**

The paper discusses in detail the results obtained and the impact of the introduced optimisations. We provide here a brief overview of the optimisations and, in the following slides, the core takeaways.



BATCH CONVERSION (IN-MEMORY)

- Improved conversion time for CSV (*2x*), JSON (*1.6x*) and XML (*>1000x*) data sources
- The *Final* release was able to convert CSV, JSON and XML datasets up to 100 MB and generating 18 M triples with the available resources

Scale	1		5		10		50		100	
Input Size (MB)	4.9		10.42		23.64		106.1		247.5	
Triples	565,489		1,800,911		3,663,380		18,009,100		36,633,800	
Release	Core	Final	Core	Final	Core	Final	Core	Final	Core	Final
CSV	22.77s	10.83s	164.95s	55.37s	544.41s	154.13s	11624.45s	3441.67s	TO	OM
JSON	50.41s	30.89s	659.11s	394.21s	2471.29s	1467.70s	66003.36s	34901.65s	TO	TO
XML	TO	16.26s	TO	123.29s	TO	434.05s	TO	12648.65s	TO	OM

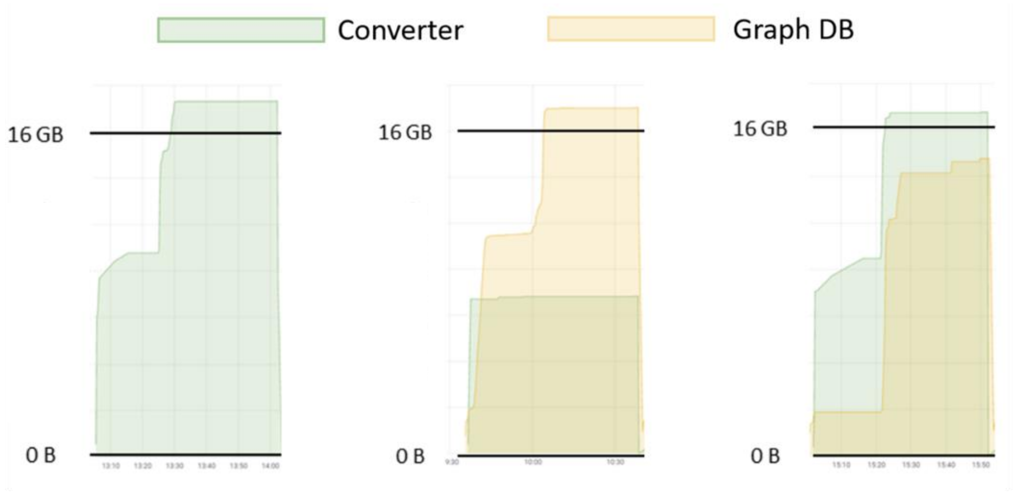
TO:
24h timeout

OM:
Out of Memory
(>24GB)

- Main factors improving performance: concurrency / multithreading, optimisation input parsing (cf. XML)
- CSV conversion time outperforms the JSON/XML one because of the impact of the libraries to access the input datasources in the lifting procedure.

BATCH CONVERSION (IN-MEMORY AND REMOTE TRIPLESTORE)

	Conversion time (s)	Lifting time (s)	Lowering time (s)	Max Mem (GB)	CPU Usage (%)
50-core-csv	11624.45	11583.49	40.97	18.84	185.56
50-final-csv	3441.67	3407.39	34.28	18.58	516.51
50-final-csv-ext	3784.34	3659.39	88.95	9.63	314.61
50-final-json	34901.65	34861.97	39.69	18.45	153.97
50-final-xml	12648.65	12614.62	34.03	18.44	540.01



50-csv

50-csv-ext

50-csv-ext
with no
incremental writes

1. Overall conversion time is **mainly influenced by the lifting phase**
2. Lifting times are influenced by the input data format
3. Lowering times are similar since the same lowering mappings are executed on the same knowledge graph.

4. Usage of external **RDF repository and incremental writes** reduces memory consumption (*2x*) but…

…it is important to take into account that an external repository implies also its own **resource usage**

5. Incremental writes should be enabled to reduce the memory consumption

Note: The conversion time when using an external repository is affected by the concurrency limitations in querying GraphDB Free

SERVICE MEDIATION

Configurations:

- *final-m-1* concurrency in lifting only at the record level
- *final-m-2* concurrency also at the triple map level

Performances:

- *140ms* for a single conversion
- 5x improvement of conversion time
- for small messages it is preferable not to introduce excessive concurrency (final-m-2 no speedup)
- very limited resource usage

Scalability:

- *100 requests/s* for a single instance of the Converter

	Conversion time (ms)	Lifting time (ms)	Lowering time (ms)	Max Mem (GB)	Max CPU Usage (%)
core-m	739	711	28	0.09	40
frel-m-1	138	107	31	0.04	25
frel-m-2	166	125	41	0.04	30

Number of concurrent requests (N)	Avg Time of processing N Requests [ms]	Interval between requests [ms]
10	131	100
50	219	20
100	775	10
150	1 663	6,7
200	1 918	5
500	3 926	2
1000	7 567	1
2500	21 114	0,4
5000	Not completely processed (queue size limit)	0,2

RECOMMENDATIONS FOR LIFTING PERFORMANCE

RML **mappings** defined for a conversion pipeline (join conditions, number of triple maps, number of logical sources, path to access the records, usage of functions…) can heavily **influence the conversion time and resource usage** of the lifting portion.

Optimizations can improve performances for the lifting, but it is **not possible to identify a single configuration performing in the best way for any lifting procedure**.

to reduce ***execution time***

- exploit concurrency in the processing of triple maps
- optimise the access to the input data sources
- leverage the same IRI generation patterns in different RML Triple Maps to avoid the usage of *join* conditions

to reduce ***memory consumption***

- implement incremental and concurrent upload of triples generated to a remote repository (shifting the performance bottleneck to the Triplestore)
- avoid using caches during the lifting procedure if not needed

Performance mainly influenced by the SPARQL queries used in the template to query the graph, and the **logic used in the template** to process the results of queries

to reduce ***execution time***

- define focused queries (e.g. avoiding *optional* clauses)
- define support data structures to access results of queries to avoid nested loops in the template logic

to reduce ***memory consumption***
remove whitespaces and newlines in the template formatting then the resulting output

Optimization to **process templates in-memory** (avoiding I/O operations) is recommended for the **service mediation** use case leading to **better execution time**.

For large batch datasets, this option should be avoided, because the template engine is able to optimize memory consumption with incremental writes to the filesystem.

RECOMMENDATIONS FOR SCALABILITY

BATCH CONVERSION: scalability of the solution is limited by memory consumption due to the materialization of large knowledge graphs

- **Virtualization techniques** can be applied for lifting, but the SOTA tools are **still not mature** enough
- **For very large datasets**, it is better to **split the conversion**:
  i. execution of the lifting procedure (if required, splitting the mappings in different executions);
  ii. bulk loading of the materialized graph(s) into the triplestore (thus avoiding incremental indexing issues on the triplestore);
  iii. on-demand lowering to convert the data to the output format when needed.
- **Different lifting tools** exist for RML and depending on the requirements they can guarantee better performances[1]

SERVICE MEDIATION: scalability of the solution is limited by the number of threads available to process the incoming concurrent requests

- (if **single** conversion pipeline) deploy more than one instance of the converter exploiting a load balancing mechanism
- (if **multiple** conversion pipelines) deploy multiple instances of a *universal converter* which is able to dynamically select and execute the relevant mappings with respect to the input/output message

[1] Arenas-Guerrero, J., Scrocca, M et al..: Knowledge graph construction with R2RML and RML: An ETL system-based overview. In: Proceedings of the 2nd International Workshop on Knowledge Graph Construction (2021), http://ceur-ws.org/Vol-2873/paper11.pdf

CONCLUSIONS

- Semantic interoperability in the transportation domain can be addressed effectively exploiting Semantic Web technologies but it is **important to address performance and scalability issues** to guarantee adoption

- Considering both the **dataset and message conversion scenario requirements**, we evaluated the Chimera framework, adopting a **declarative approach** based on **RML mappings for lifting** and on **Apache Velocity templates with SPARQL queries** for lowering.

- We managed to generate and handle **large knowledge graphs**, and we proved robustness and low conversion times with hundred of **concurrent messages** per second.
- We defined a set of **recommendations** to improve performance and scalability of the approach in different scenarios.
- As **future work**, we would like to:
  - investigate and implement in Chimera further optimisations for the lifting procedure,
  - setup a more comprehensive and structured benchmark for the message conversion scenario (e.g., using GTFS-RT data).

Thank you

for your attention!

Mario Scrocca
mario.scrocca@cefriel.com