This project has received funding from the European's Union Horizon 2020 research innovation programme under Grant Agreement No. 957258

Architecture for Scalable, Self-*, human-centric, Intelligent, Secure, and Tactile next generation IoT

ASSIST-IoT Technical Report #16

RiverBench: an Open RDF Streaming Benchmark Suite

Piotr Sowinski1, Maria Ganzha1 and Marcin Paprzycki

RiverBench: an Open RDF Streaming Benchmark Suite

Piotr Sowiński[1,2][0000−0002−2543−9461], Maria Ganzha[1,2][0000−0001−7714−4844], and Marcin Paprzycki[2][0000−0002−8069−2152]

[1] Warsaw University of Technology, Warsaw, Poland
piotr.sowinski.dokt@pw.edu.pl, maria.ganzha@pw.edu.pl

[2] Systems Research Institute, Polish Academy of Sciences, Warsaw, Poland
{firstname.surname}@ibspan.waw.pl

Abstract. RDF data streaming has been explored by the Semantic Web community from many angles, resulting in multiple task formulations and streaming methods. However, for many existing formulations of the problem, reliably benchmarking streaming solutions has been challenging due to the lack of well-described and appropriately diverse benchmark datasets. Existing datasets and evaluations, except a few notable cases, suffer from unclear streaming task scopes, underspecified benchmarks, and errors in the data. To address these issues, we firstly systematize the different RDF data streaming tasks in a clear taxonomy and outline practical requirements for benchmark datasets. We then propose RiverBench, an open and collaborative RDF streaming benchmark suite that applies these principles in practice. RiverBench leverages continuous, community-driven processes, established best practices (e.g., FAIR), and built-in quality guarantees. The suite distributes datasets in a common, accessible format, with clear documentation, licensing, and machine-readable metadata. The current release includes a diverse collection of non-synthetic datasets generated by the Semantic Web community, representing many applications of RDF data streaming, all major task formulations, and emerging RDF features (RDF-star). Finally, we present a list of research applications for the suite, demonstrating its versatility and value even beyond the realm of RDF streaming.

Keywords: RDF streaming · Benchmark · RDF stream models · FAIR · RDF-star.

1 Introduction

Over the years, the concept of RDF streams was investigated from many angles, resulting in streaming serializations, querying and reasoning engines, programming frameworks, datasets, and other resources [37,35,38,29]. However, the landscape of RDF streaming is fragmented, with many different, often informal task formulations and incompatible solutions. Benchmarking RDF streaming systems is still, in many cases, challenging, with a clear lack of well-described and appropriately diverse benchmark datasets. Although some types of systems can be benchmarked reliably (e.g., streaming query engines), these benchmarks hardly generalize to other types of systems or task formulations.

To address these issues, in this work we make a two-fold contribution. Firstly, we systematize RDF streaming task formulations in a clear taxonomy and outline practical requirements for streaming benchmark datasets. Secondly (and primarily), we introduce **RiverBench**, an open and collaborative RDF streaming benchmark suite that covers all major task formulations, emerging RDF features (RDF-star), and is applicable even beyond the realm of RDF streaming. RiverBench puts the proposed task taxonomy and requirements to practice, aiming to be a sustainable, community-driven hub for streaming RDF datasets.

The paper is structured as follows. Section 2 provides background and detailed motivation for the work. Section 3 outlines the theoretical foundations for RiverBench. Section 4 describes the structure and mechanisms of the suite. Section 5 lists the datasets included in the current release. Section 6 presents the potential applications of the suite. Section 7 describes the approach to collaborative development and the sustainability plan. Section 8 presents the conclusions.

2 Background

This section summarizes the current state of the art with regard to RDF streaming task formulations and streaming RDF benchmarks. The aim of the summary is to provide a background for the theoretical discussion in section 3, as well as motivation for why RiverBench was created and for each of its design decisions.

2.1 RDF Streaming Task Formulations

RDF streaming is a term that over the years was applied to many, often quite dissimilar tasks. Here, we present the most important and representative works that investigated this matter.

The first group of articles focus on streaming protocols for transmitting RDF data. One of the earlier examples is Streaming HDT (S-HDT) [20], which facilitates streaming RDF data in IoT devices. More specifically, producers (IoT devices) emit a stream of compressed, unnamed RDF graphs. RDSZ [14] is a later method that also attempts streaming compressed RDF graphs over the network. ERI [12] is a similar approach that uses a slightly different definition of the problem: *ERI considers an RDF stream as a continuous flow of blocks (with predefined maximum size) of triples.* In fact, on the high level, ERI streams a continuous sequence of triples that is then divided into discrete elements (RDF graphs) and transmitted over the network. Therefore, it can be argued that ERI can serve two formulations of the streaming problem. The RDF EXI communication protocol for constrained devices [22] also focuses on streaming RDF graphs. Finally, the recent Jelly protocol [35] has a task formulation similar to ERI, of streaming a sequence of RDF statements that are divided into elements. The main difference is that Jelly can also stream quad statements, making it generalizable to streaming any RDF datasets.

The second group of papers revolves around the work of the RDF Stream Processing Community Group (RSP), which mostly focuses on timestamped streams of RDF data, i.e., streams in which elements are annotated with a time instant or interval [37]. Here, RSP-QL [9] is an important work, as it is a rare example of a full formalization of RDF stream semantics. In RSP-QL, an RDF stream is a potentially unbounded sequence of timestamped triple statements (timestamp-triple tuples), in non-decreasing time order. However, the landscape of RSP implementations is diverse and has not converged yet to single, unifying semantics. The recent RSP4J [37] defines an RDF stream as an infinite sequence of three-tuples: *RDF object*, event time, processing time. The paper is missing however an explanation of what is an *RDF object*, and we assume from context it to be an RDF graph. The RSP Community Group has also worked on a draft specification for the RSP Data Model [33]. In the draft (that does not represent the group's consensus), an RDF stream is a sequence of *timestamped graphs* called its elements. Each timestamped graph is an RDF dataset with exactly one named graph pair $\langle n, G \rangle$, where G is an RDF graph and n is an IRI or a blank node. The dataset may include any number of triples in the default graph, but there must be exactly one timestamp triple $\langle n, p, t \rangle$, where p is the designated timestamp property and t is the timestamp. Although this definition was not agreed upon yet, it is notably well-grounded in the RDF specification.

The last broad category of RDF streaming solutions are those that process simple sequences of RDF statements (triples or quads). Two examples include the RDF4J Rio toolkit [10] and Apache Jena's RIOT [2], both of which can parse or serialize streams of statements, reading from or writing to a Java byte stream. The two most obvious serializations for this task are N-Triples and N-Quads, however, Jena supports also *streamed block formats* for Turtle and Trig, where statements are transmitted in small batches (blocks). The aforementioned RDSZ, ERI, and Jelly methods can also be used for this task, by splitting the statement stream into discrete elements.

From the above it is evident that there is no common model of RDF streams. Furthermore, each listed work motivates the choice of its formulation well, by following the needs of a specific use case.

2.2 RDF Streaming Benchmarks

Several RDF streaming benchmarks and datasets were proposed in the past, often tailored to a specific task formulation. In the streaming protocol-oriented research, the first benchmark was published with RDSZ [14], later expanded in the ERI paper [12]. The ERI benchmark[3] consists of 16 datasets of various lengths and domains. The paper presents several statistics about the datasets, which helps with evaluating a method's performance on a given dataset. Each dataset is a flat N-Triples file and thus the stream is not divided into discrete elements. Instead, in the evaluations a fixed element size was used (e.g., 4096 triples), which is often not representative of the use cases, as in reality element size

[3] http://rohub.linkeddata.es/RO-ISWC-14/

can vary even message-to-message. Secondly, the benchmark is skewed towards some applications or ontologies, making it less representative of real workloads. For example, 5 of the datasets focus on weather measurements, and 3 of those (LOD_*) follow the same schema. Thirdly, some of the datasets contain syntax errors [35], which hampers their reuse. Finally, the license and provenance of each dataset is not clearly stated, making it hard to modify and republish them.

Many benchmark datasets and tools were developed for evaluating RSP reasoning and querying engines – SRBench [42], LSBench [26], CityBench [1], RSPLab [39], WatDiv [17], Stream Reasoning Playground [34], and more. These works have largely focused on evaluating the efficiency of query engines, and not on the datasets themselves, therefore, many have used data generators for a single type of stream. These generators tend to be tightly integrated with the specific frameworks or streaming models that are evaluated, and thus are very hard (if at all possible) to reuse outside their context. Secondly, synthetic datasets are a reasonable choice for some evaluations, but in others less so (such as compression efficiency evaluations). Here, notably, SRBench uses a real-world, streaming sensor dataset – LinkedSensorData [30]. CityBench also uses several real-world datasets from the CityPulse project [1]. The datasets of SRBench and CityBench are clearly licensed and can be obtained in standard RDF serializations. However, they lack an explicit split into stream elements.

3 Theoretical Foundations

This section lays down the theory that guides the design of RiverBench. Here, we propose a taxonomy of RDF streaming task formulations based on past research, and a set of practical requirements for benchmark datasets.

3.1 Task Formulation

In what follows, we propose a set of semi-formal definitions for all prominent streaming task formulations, as well as their taxonomy. As these tasks are based on past research, the main contribution here is their systematization and semi-formalization. It should be noted that a full, strict formalization of RDF streaming goes well beyond the scope of this work and remains an open research topic. The basis for all following definitions is the RDF 1.1 W3C Recommendation [7].

Definition 1. *An **(elementwise) RDF stream** is an ordered, potentially unbounded sequence of RDF stream elements.*

Although we assume that streams can be unbounded, RiverBench focuses on real-world datasets and thus all of its datasets are finite. Secondly, the assumption about ordering of stream elements may be relaxed in some applications, depending on the used Quality of Service settings and other factors [24].

Definition 2. *An **RDF stream element** is a packet of semantic data that must be transmitted and processed atomically. An element's semantics are maintained only if it is processed as a whole.*

Here, the atomicity only applies to the semantic, RDF layer of the stream, i.e., the entire element must be parsed as a single logical unit. However, this does not preclude the use of, e.g., packet chunking in the network protocol.

The first two definitions are very general and by themselves cannot be applied in practice. Hence, we define four concrete types of RDF streams.

Definition 3. *An **RDF triple stream** is an RDF stream in which every element is an unnamed (default) RDF graph.*

In other words, an element of a triple stream is a set of triples. This definition corresponds to the how several streaming methods function, including ERI, RDSZ, S-HDT, RDF EXI, and Jelly.

Definition 4. *An **RDF quad stream** in an RDF stream in which every element is an RDF dataset.*

Similarly, an element of a quad stream is a set of quads. This definition is very broad, as it allows for streaming any kind of RDF data, but it is also scarcely discussed in literature, with Jelly being a rare implementation of this concept.

Definition 5. *An **RDF graph stream** is an RDF stream in which every element is an RDF dataset. In the dataset there must be exactly one named RDF graph pair $\langle n, G\rangle$, where G is an RDF graph, and n is the graph node. Apart from graph G, the dataset may contain any number of triples in the default graph.*

Definition 6. *A **timestamped RDF graph stream** is an RDF graph stream in which every element's default graph includes exactly one timestamp triple $\langle n, p, t\rangle$, where p is the designated timestamp property and t is the timestamp.*

It can be observed that all graph streams are quad streams, raising the question for why a separate definition is needed. This is primarily because graph streams provide an intuitively important distinction, in that every element corresponds to exactly one named RDF graph with some optional information about this graph. Secondly, similar problem statements are common in the research of the RSP community [37]. This formulation also allows us to define *timestamped streams*, where each graph is associated with a timestamp. This particular definition is designed to be compatible with the draft RSP Data Model [33].

Additionally, we define a flat sequence of statements (a flat RDF stream).

Definition 7. *A **flat RDF stream** is an ordered, potentially unbounded sequence of RDF statements (either triples or quads).*

A flat stream can be represented simply as an N-Triples or N-Quads file. The reason for including this definition is that it is trivial to derive a flat RDF stream from any of the RDF stream variants defined above, by concatenating all stream elements. Flat streams can also be used for many non-streaming applications, such as benchmarking compression algorithms (see section 6). Moreover, this task formulation is also used for some streaming applications, and several streaming benchmarks were distributed as flat streams (e.g., ERI [12]).

The presented definitions form a taxonomical structure of task formulations, summarized in Figure 1. In RiverBench, the only assumptions about the specification of the task stem from these definitions. Notably, there are no assumptions about the involved actors, network protocols, serialization methods, or Quality of Service settings, as they vary greatly depending on the use case. Although this matter is beyond the scope of this work, our recommendation is that to ensure high quality and reproducibility of benchmark results, these and other parameters should be systematically observed, noted, and reported in evaluations. Further discussion on this matter can be found in relevant literature [1,23,24].

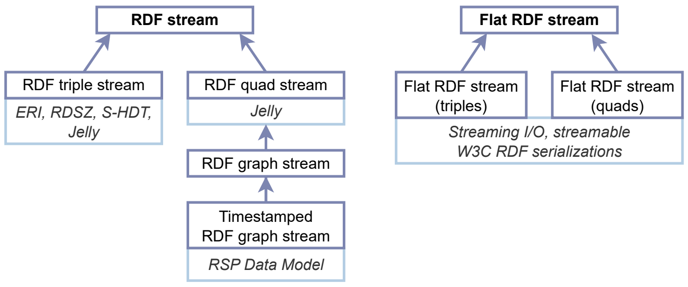


Fig. 1. Taxonomy of RDF streaming tasks. The italicized annotations indicate some of the most closely related methods or concepts.

3.2 Requirements for Benchmark Datasets

Based on past research, observed issues in reproducibility, and established best practices (such as the Findability, Accessibility, Interoperability, and Reuse principles – FAIR [41]), we propose a set of practical requirements for streaming RDF datasets to be used in benchmarks and other performance evaluations. These requirements are meant to ensure that the obtained results are representative of real-world performance of the method, and that they can be easily reproduced.

R1: The datasets must be varied with regard to technical and non-technical aspects. The range of represented use cases, ontologies, and technical features should be as wide as possible.
Rationale: The suite should be varied to be representative of a wide range of use cases, and to avoid skew towards any use case or technical aspect.

R2: The datasets must be freely accessible and packaged in a common, easy-to-use format.
Rationale: Ease of access is crucial to ensuring that the datasets will be systematically reused in future evaluations by researchers [41,15].

R3: The datasets must be well-described with documentation and machine-readable metadata. All relevant information about the technical and non-technical characteristics of the dataset must be included.

Rationale: Information about the dataset's use case, technical characteristics, and more are often crucial to understanding why a method performs well or poorly with a given dataset [12]. Machine-readable metadata can also help automate some evaluation tasks, or serve as a source of information for research on the datasets themselves.

R4: The license of the dataset and its metadata must be free and explicitly stated. The attribution of the source must be clear. The license must not prohibit commercial use or modification.
Rationale: Clear licensing ensures that the datasets can be freely reused in evaluations. The licenses must be permissive so that the datasets can be adapted, republished, and used in a wide variety of contexts.

R5: The datasets must be semantically versioned and the past versions must be accessible under a persistent URL (PURL).
Rationale: This is a crucial requirement for ensuring reproducibility, as different versions of datasets will produce different, incomparable results.

R6: The datasets must be explicitly divided into discrete elements (if applicable to the given problem).
Rationale: Previous RDF streaming benchmarks did not always specify the division into elements explicitly [12,14], which allowed different methods to use different element sizes, making the approaches difficult to compare. This requirement does not apply to the flat RDF stream task formulation.

R7: The length of stream datasets must be at least 10,000 elements.
Rationale: Longer streams provide more reliable benchmark results, reducing the effects of random noise, cache *warm-up*, and other factors.

R8: The datasets must be valid RDF that can be parsed easily.
Rationale: Past benchmark datasets sometimes included errors [35], hampering their reuse. This requirement calls for *valid RDF* specifically, but, if clearly marked as such in metadata, distributing datasets using non-standard RDF features should also be possible.

4 Structure of RiverBench

This section presents the overall structure of the RiverBench benchmark suite, which puts the above theory to practice. The suite from its inception has been envisioned as an open, community-driven, and collaborative project, that follows FAIR principles and other established best practices. These crucial aspects have influenced every part of the suite, as described in detail below.

4.1 Datasets

Datasets are added to RiverBench through a documented, public process that ensures that the relevant requirements are met (as proposed in the previous section). Firstly, the contributor fills out an issue template on GitHub, giving information about licensing, attribution, used task formulation, technical characteristics (e.g., stream length), the original use case, and more (**R1**, **R4**, **R6**,

R7). An administrator reviews the template, asking for clarification if needed, validates that the requirements are met, and creates a public repository for the dataset. The contributor then uploads the dataset's source files and detailed metadata in a Turtle file. For both tasks, clear documentation and assistance is provided. The Turtle metadata file only includes information that must be specified manually (e.g., license, description, topics), and follows a predefined template. The rest of the process is fully automated, with Continuous Integration / Continuous Deployment (CI/CD) jobs checking the validity of the data, packaging the dataset, adding more metadata, generating documentation pages, and uploading the final packages. The dataset is validated with two RDF libraries (RDF4J and Apache Jena), using the strictest parsing settings (**R8**). Additional automatic checks ensure that the dataset's content is consistent with its metadata and the specified streaming task formulation. The full procedure of adding a new dataset is described in the documentation[4]. A visual summary of the process is presented in Figure 2. More details on the metadata, packaging, and publishing steps can be found in the following subsections.

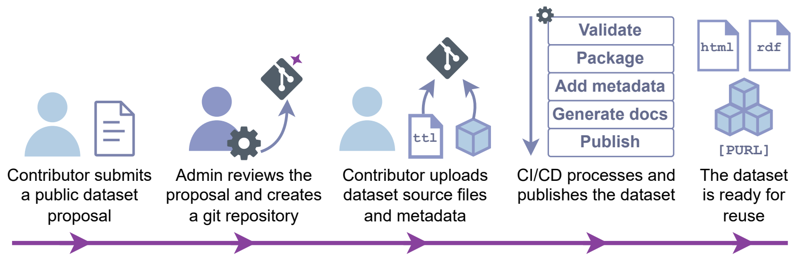

Validate
Package
Add metadata
Generate docs
Publish
html
rdf
ttl
[PURL]
Contributor submits a public dataset proposal
Admin reviews the proposal and creates a git repository
Contributor uploads dataset source files and metadata
CI/CD processes and publishes the dataset
The dataset is ready for reuse

Fig. 2. The process of adding and publishing a dataset in RiverBench.

By being completely public and open, the process is designed to invite external contributions from the community. It can also evolve over time, following community feedback (see section 7). A key innovation over previous benchmarks is the inclusion of automated CI/CD jobs that guarantee that each dataset follows the same sequence of validation, packaging, and publishing steps.

4.2 Benchmark Profiles

RiverBench covers all streaming task formulations presented in section 3.1. However, in most applications, datasets representing different formulations should not be mixed. Thus, RiverBench employs a clear mechanism of grouping datasets into *profiles*, with each profile corresponding to a specific task formulation and supported RDF features. The profiles also help research reproducibility, by being well-defined collections of datasets. Researchers can simply share the name

[4] https://w3id.org/riverbench/documentation/creating-new-dataset

and version (or the PURL) of the profile they used in the evaluation, to specify which datasets were involved. There are 7 base profiles in the current release of RiverBench, corresponding to different streaming task formulations:

– **stream-triples** – RDF triple streams.
– **stream-graphs** – RDF graph streams.
– **stream-quads** – RDF quad streams, including graph streams.
– **stream-mixed** – all RDF streams, including quad and triple streams.
– **flat-triples** – flat RDF triple streams.
– **flat-quads** – flat RDF quad streams.
– **flat-mixed** – all flat RDF streams, including quad and triple flat streams.

Each of the base profiles has 4 variants that differ in the RDF features that are used in the datasets:

– **(no suffix)** – fully RDF 1.1-compliant datasets, with no extra features.
– ***-rdfstar** – datasets may use RDF-star, but must remain compliant with the RDF-star draft specification [3].
– ***-nonstandard** – datasets may contain generalized triples and datasets, or other non-standard features, but not RDF-star.
– ***-rdfstar-nonstandard** – datasets may use RDF-star and other non-standard features.

Thus, there are in total 28 profiles in the current release[5], which form a taxonomy that is reflected in the metadata and the documentation. This taxonomy not only helps organize the profiles, but also allows for making partial comparisons between evaluations made with different profiles. For example, if method A supports only streaming graphs and was evaluated with profile `stream-graphs`, it can be compared to method B that used the more general `stream-quads-rdfstar` profile, as all datasets of the former are in the latter.

The conditions for datasets to be included in a profile are specified semantically in the profile's metadata file (in Turtle) and resolved by the CI automatically against the metadata of the datasets. Then, the CI generates the needed semantic relations and documentation. Creating a profile boils down to writing down its conditions, the rest is handled automatically. The profiles are also continuously updated whenever changes in datasets are made. The set of profiles can evolve over time to meet the community's needs, as described in section 7.

4.3 Metadata

Each dataset and profile in the suite has associated machine-readable RDF metadata describing its technical and non-technical aspects (**R3**). For interoperability, the metadata primarily uses the Data Catalog Vocabulary (DCAT) version 3 [6] and Dublin Core [8], with some additional properties from the FOAF and SPDX vocabularies. To represent the specific semantics of RiverBench, DCAT

[5] https://w3id.org/riverbench/profiles

was extended with additional properties and classes. This additional ontology, although playing a secondary role in the suite, is fully documented and published under a PURL[6], using best practices developed by the community [19], achieving 92% score in *FOOPS!* [18] and 4/4 stars in DBpedia Archivo [16].

Part of the metadata is written manually, but a large portion is generated automatically during the packaging process. The auto-generated part includes rich metadata of the packages and dataset statistics. What follows is a high-level overview of the metadata that is included with each dataset.

Non-technical Metadata

- **Description** – information about the broader context in which the dataset was used or generated.
- **License and attribution** – licensing and authorship information of the dataset, with a link to the license's definition in the SPDX License List [4].
- **Topics / themes** – tags selected from a pre-defined SKOS taxonomy, describing the dataset's application area and the type of included data.

Technical Metadata

- **Used ontologies** – a list of the most prominently used ontologies in the dataset.
- **Stream element type** – possible values: *triples*, for RDF triple streams; *quads*, for RDF quad streams; *graphs*, for RDF graph streams.
- **Stream element split type** – how was the dataset divided into stream elements. The types can be combined and include: *statement count*, where the stream was split into elements of arbitrary length; *topic*, where each element contains information about a different subject; *time*, for elements associated with a particular point in time or a time interval. Additional details on the split can also be provided.
- **Temporal property IRI** – used only for streams with a time-based element split, indicates the RDF property that marks the timestamps of stream elements. It is also the time property in timestamped RDF graph streams.
- **Stream element count** – total number of stream elements in the dataset.
- **RDF features used** – a checklist of major standard and non-standard RDF features used in the dataset.
- **Distributions (packages)** – information about the available packaged variants of the dataset (see section 4.4), including byte size, checksums, content types, download URLs, and more.
- **Statistics** (for each distribution) – automatically calculated statistical parameters (mean, standard deviation, minimum, maximum, sum) about several variables in the population of stream elements. The variables include the count of: IRIs, blank nodes, literals (simple, datatype, language), quoted triples (RDF-star), subjects, predicates, objects, graphs, and statements.

[6] https://w3id.org/riverbench/schema/metadata

The metadata can be downloaded via links on the HTML documentation pages, as well as using the HTTP content negotiation mechanism on their PURLs (supported formats are: RDF/XML, N-Triples, and Turtle). The metadata is comprehensive, with the hope of enabling quantitative research on the effect of different types of datasets on systems processing RDF. It is released under a permissive license (CC BY 4.0) and can be freely reused for any purpose (**R4**). The metadata can be easily expanded, with dataset contributors being able to attach any additional semantic information to their datasets. The automatically generated information can also be extended with ease, for the entire suite.

4.4 Packaging and Publishing

The datasets are packaged by the CI into files in a common format based on established standards (**R2**). The source archives uploaded by contributors are processed by the CI in a streaming manner, making the process scalable to even very large datasets. Each stream element is parsed, validated, and re-serialized. For every dataset two types of distributions are made: the (elementwise) streaming distributions and the flat distributions (flat RDF streams). For both types, several stream length variants are prepared, starting with 10,000 elements, and increasing by a factor of 10 with each step. The length variants can be useful when testing methods that do not require very large datasets, or when streams of equal length are needed for a fair comparison.

In RDF streams, elements are serialized in W3C-standard Turtle or TriG, depending on the type of the stream. If the dataset uses RDF-star, the Turtle-star and TriG-star formats [3] are used. Stream elements are stored as separate, sequentially numbered files in a .tar.gz archive, which is a common, interoperable, and streamable format. The files in the archive are laid out sequentially, allowing the package to be processed in a streaming manner (one element at time, without decompressing the whole archive). Flat RDF streams are simply serialized as either N-Triples or N-Quads files, or, in case RDF-star is used, N-Triples-star and N-Quads-star. The serialized data is gzip-compressed. This format is also streamable, as the file can be decompressed on-the-fly and read line-by-line.

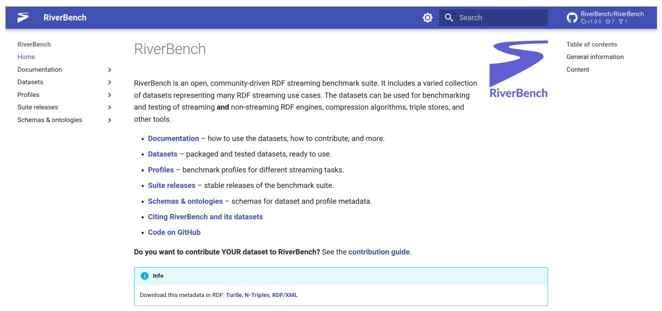
These formats were chosen due to them being widely supported, ensuring the accessibility of the datasets. They were also specifically designed to be as simple as possible, while maintaining good performance characteristics. It should be noted that there are currently no standardized or widely supported formats for exchanging RDF streams. Distributing the datasets in such a dedicated format, e.g., Jelly [35] (in addition to the current ones) will be investigated in the future. Full documentation of the packaging formats can be found on the website[7].

Packaged datasets are published along with their metadata and documentation under the RiverBench PURL. The releases use a versioning scheme derived from Semantic Versioning [32], giving clear compatibility guarantees between different releases and allowing users to refer to a specific stable version (**R5**). The profiles and the entire suite are also semantically versioned.

[7] https://w3id.org/riverbench/documentation/dataset-release-format/

4.5 Website and Documentation

RiverBench's publicly available website gathers all information about the suite's datasets, profiles, documentation, and licensing (**R3**). Dataset and profile documentation is generated automatically from the metadata, ensuring that all information is human-readable and accessible. Documentation for older versions of datasets and profiles is preserved for future reference (**R5**). The website is hosted under a PURL registered with w3id.org. Under the same base PURL, machine-readable metadata for all resources is available (in RDF), as well as download links to dataset packages. Figure 3 presents the main page of the website.



RiverBench

RiverBench is an open, community-driven RDF streaming benchmark suite. It includes a varied collection of datasets representing many RDF streaming use cases. The datasets can be used for benchmarking and testing of streaming **and** non-streaming RDF engines, compression algorithms, triple stores, and other tools.

- **Documentation** – how to use the datasets, how to contribute, and more.
- **Datasets** – packaged and tested datasets, ready to use.
- **Profiles** – benchmark profiles for different streaming tasks.
- **Suite releases** – stable releases of the benchmark suite.
- **Schemas & ontologies** – schemas for dataset and profile metadata.
- **Citing RiverBench and its datasets**
- **Code on GitHub**

Do you want to contribute YOUR dataset to RiverBench? See the **contribution guide**.

Info

Download this metadata in RDF: **Turtle**, **N-Triples**, **RDF/XML**

Fig. 3. Screenshot of the main page of the RiverBench website.

4.6 FAIR Compliance – Summary

The following is a summary of how RiverBench complies with the FAIR principles [41]. The listed mechanisms are applied systematically and uniformly to every resource (dataset, profile, etc.) in RiverBench and enforced with pre-defined community processes and CI jobs.

F1: All of RiverBench's resources are assigned a globally unique and persistent URL (under https://w3id.org/riverbench).

F2: RiverBench's resources are described with rich metadata with links to external sources of information.

F3: Metadata of resources clearly indicates the identifiers of the data, using appropriate RDF properties.

F4: RiverBench's website is indexable by search engines and serves as a hub for accessing other resources. Additionally, RiverBench is archived in Zenodo, and its ontologies are registered in DBpedia Archivo [16] and Linked Open Vocabularies [40].

A1: The resources are easily accessible via standardized protocols (HTTPS) and mechanisms (content negotiation).

A2: The entirety of RiverBench, including its metadata, is archived in Zenodo. See also the sustainability plan in section 7.

I1: RiverBench uses RDF 1.1 and standard serializations for its metadata.
I2: The metadata of resources uses established vocabularies (DCAT, Dublin Core, FOAF), as well as a custom ontology extension that follows the FAIR principles as well.
I3: The resources in RiverBench are interlinked through well-defined links in their metadata. They also include references to external sources of information, such as SPDX licenses or ontologies used in datasets.
R1: Dataset metadata includes rich contextual information about the context in which the dataset was created. The metadata of resources is plentiful, including, for example, detailed statistics about each dataset distribution.
R1.1: All resources are clearly and permissively licensed.
R1.2: All resources have human- and machine-readable authorship and provenance information associated with them.
R1.3: RiverBench makes a best effort to implement best practices for metadata developed by the Semantic Web community, as presented above.

5 Datasets Included in the Current Release

At the time of submission, the current version of RiverBench is 1.0.0, featuring 12 datasets[8], of which 9 are RDF triple streams, 2 are graph streams, and 1 is a quad stream. All datasets were added by the authors using the public community process described in section 4.1, which can be reviewed in the suite's issue tracker. The authors had direct involvement in only two of the datasets (from ASSIST-IoT), with the rest being gathered from community use cases.

- assist-iot-weather & assist-iot-weather-graphs – two variants (an RDF triple stream and a graph stream) of the same source dataset – SOSA/SSN weather measurements collected in the ASSIST-IoT project [36].
- citypulse-traffic & citypulse-traffic-graphs – two variants (an RDF triple stream and a graph stream) of the road traffic dataset from the CityPulse project, also used in CityBench [1].
- dbpedia-live – high-volume triple stream from the DBpedia Live service, describing changes in Wikipedia [27].
- digital-agenda-indicators – RDF triple stream of very regular statistical information about the European information society [11].
- linked-spending – RDF triple stream, part of the LinkedSpending dataset, which collects government spending statistics from around the world [21].
- lod-katrina – RDF triple stream, the Katrina weather measurement dataset from LinkedSensorData [26]. Also used by SRBench [42] and ERI [12].
- muziekweb – RDF triple stream, a Dutch knowledge base about music [28].
- nanopubs – RDF quad stream, a set of freely-licensed Nanopublications, small units of publishable information [25].
- politiquices – RDF triple stream describing news articles in Portuguese and the presented political stances [5].

[8] https://w3id.org/riverbench/datasets

– yago-annotated-facts – RDF-star triple stream, a subset of YAGO 4 [31] including only fact annotations in RDF-star.

In the current release there is one RDF-star dataset, three non-triple datasets, and no datasets that would use non-standard RDF features beyond RDF-star. The main reason for this is that very few such datasets were publicly released and the authors had limited resources to prepare the datasets themselves. Nonetheless, RiverBench is to our knowledge the most technically diverse RDF streaming benchmark to date, and its collection of datasets is expected to expand in future releases with the help of the Semantic Web community. We have deliberately avoided adding more similar datasets to the suite (e.g., other datasets from LinkedSensorData or CityBench), to keep RiverBench as diverse as possible.

6 Applications

RiverBench, by addresssing many task formulations, has broad application potential. This section presents the envisioned research applications of the suite.

– Benchmarking streaming RDF protocols, such as ERI [12], S-HDT [20], and Jelly [35]. The `stream-*` profiles can be used for this purpose.
– Benchmarking RDF parsers and serializers (streaming and non-streaming), on a wide variety of datasets (including RDF-star).
– Evaluating compression performance of compact RDF representations, both streaming (ERI, S-HDT, Jelly, etc.) and non-streaming (e.g., HDT [13], with the `flat-*` profiles).
– Serving as a basis for streaming and non-streaming reasoning and querying benchmarks. For this, additional work of defining the queries and/or reasoning rules would be needed.
– Stress-testing RDF processing systems with representative, real-world data.
– Testing compatibility of RDF processing systems with real datasets that may contain uncommon edge cases (e.g., large literals, complex graphs).
– Conducting research on the streaming datasets, their complexity, applications, performance characteristics, etc.

More applications are expected to be identified over time, and RiverBench will welcome requests for enhancements that can help broaden its audience.

7 Collaborative Development and Sustainability Plan

To remain representative of the community's evolving needs, RiverBench must facilitate collaboration of various stakeholders from the industry and academia. Therefore, it takes a thoroughly open approach, with its every aspect being fully public and permissively licensed (metadata and documentation under CC BY 4.0, code under Apache 2.0). The main hub of activity is the RiverBench GitHub organization, which includes the source code and the issue tracker, freely accessible to any contributor. A comprehensive contributor's guide was prepared[9], which, along with detailed documentation, explain how to contribute new datasets, improve metadata and documentation, and modify the code.

Sustainability Plan. Making RiverBench sustainable was one of the primary considerations from the start. RiverBench does not incur *any* infrastructure maintenance costs, due to it being hosted exclusively on the permanently free and well-established infrastructure of GitHub and w3id.org. In case of, for example, a dramatic change to GitHub's open source project policy, the project requires for its basic needs only static file hosting and can be moved easily to a different service by the virtue of PURLs. RiverBench's source code and datasets are also archived independently in Zenodo. Regarding labor costs, the project aims to attract a wide audience, by being applicable in many scenarios. It is hoped that this will allow RiverBench to build a stable community that will be able to maintain it sustainably. Moreover, the vast majority of processes in the suite are fully automated, therefore, the workload needed to maintain it is minimal.

8 Conclusion

In this paper, we tackle the relevant issue of the lack of datasets for evaluating the very diverse RDF streaming solutions. We systematize streaming tasks in a clear taxonomy, outline the requirements for streaming benchmark datasets, and introduce RiverBench, an open RDF streaming benchmark suite that applies these principles in practice. We hope that the suite, having the values of FAIR, community collaboration, and sustainability at its core, will be welcomed by the community as a useful resource. We aim to continue improving RiverBench in the future, especially by expanding its coverage of less-common streaming tasks, improving its metadata, and its tooling. The possible future enhancements can be discussed (and new ones can be proposed) in the project's issue tracker.

We invite all researchers and practitioners in the fields of Semantic Web and Knowledge Graphs to collaborate on future versions of RiverBench, to make it closely aligned with the community's needs. Especially welcome will be new datasets from varied application areas. We hope that with future community-led efforts, the suite will greatly improve in terms of quality, size, and diversity.

Resource Availability Statement:

– Website of RiverBench: https://w3id.org/riverbench
– Source code for RiverBench and its tools: https://github.com/RiverBench
– Public issue tracker: https://github.com/RiverBench/RiverBench/issues
– Archive / backup in Zenodo: https://doi.org/10.5281/zenodo.7909063

Acknowledgements This work was partially funded from the EU's Horizon 2020 research and innovation programme under grant agreement No 957258, in the ASSIST-IoT project.

[9] https://w3id.org/documentation/contribute/

References

1. Ali, M.I., Gao, F., Mileo, A.: CityBench: A configurable benchmark to evaluate RSP engines using smart city datasets. In: The Semantic Web-ISWC 2015: 14th International Semantic Web Conference, Bethlehem, PA, USA, October 11-15, 2015, Proceedings, Part II. pp. 374–389. Springer (2015)
2. Apache Software Foundation: Working with RDF streams in Apache Jena (2023), https://jena.apache.org/documentation/io/streaming-io.html, accessed on May 7, 2023
3. Arndt, D., Broekstra, J., DuCharme, B., Lassila, O., Patel-Schneider, P.F., Prud'hommeaux, E., Ted Thibodeau, J., Thompson, B.: RDF-star and SPARQL-star. Draft community group report, W3C RDF-DEV Community Group (Dec 2022), https://w3c.github.io/rdf-star/cg-spec/editors_draft.html
4. ASPDX Workgroup, a Linux Foundation project: SPDX license list (Feb 2023), https://spdx.org/licenses/, accessed on May 6, 2023
5. Batista, D.S.: Politiquices (2022), https://www.politiquices.pt/, accessed on May 7, 2023
6. Cox, S., Winstanley, P., Browning, D., Albertoni, R., Perego, A., Beltran, A.G.: Data Catalog Vocabulary (DCAT) – version 3. W3C working draft, W3C (Mar 2023), https://www.w3.org/TR/2023/WD-vocab-dcat-3-20230307/
7. Cyganiak, R., Wood, D., Lanthaler, M.: RDF 1.1 concepts and abstract syntax. W3C recommendation, W3C (Feb 2014), https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/
8. DCMI Usage Board: DCMI metadata terms. Tech. rep., DCMI (Jan 2020), https://www.dublincore.org/specifications/dublin-core/dcmi-terms/
9. Dell'Aglio, D., Della Valle, E., Calbimonte, J.P., Corcho, O.: RSP-QL semantics: A unifying query model to explain heterogeneity of RDF stream processing systems. International Journal on Semantic Web and Information Systems (IJSWIS) **10**(4), 17–44 (2014)
10. Eclipse Foundation, Inc.: Parsing and writing RDF with Rio (2023), https://rdf4j.org/documentation/programming/rio/, accessed on May 7, 2023
11. European Commission, Directorate-General for Communications Networks, Content and Technology: Key indicators – digital scoreboard – data & indicators (2023), https://digital-agenda-data.eu/datasets/digital_agenda_scoreboard_key_indicators, accessed on May 7, 2023
12. Fernández, J.D., Llaves, A., Corcho, O.: Efficient RDF interchange (ERI) format for RDF data streams. In: International Semantic Web Conference. pp. 244–259. Springer (2014)
13. Fernández, J.D., Martínez-Prieto, M.A., Gutiérrez, C., Polleres, A., Arias, M.: Binary RDF representation for publication and exchange (HDT). Journal of Web Semantics **19**, 22–41 (2013)
14. Fernández, N., Arias, J., Sánchez, L., Fuentes-Lorenzo, D., Corcho, Ó.: RDSZ: an approach for lossless RDF stream compression. In: European Semantic Web Conference. pp. 52–67. Springer (2014)
15. Frey, J., Streitmatter, D., Arndt, N., Hellmann, S.: Reproducibility crisis in the LOD Cloud? studying the impact of ontology accessibility and archiving as a counter measure. In: The Semantic Web–ISWC 2022: 21st International Semantic Web Conference, Virtual Event, October 23–27, 2022, Proceedings. pp. 91–107. Springer (2022)

16. Frey, J., Streitmatter, D., Götz, F., Hellmann, S., Arndt, N.: DBpedia Archivo: a web-scale interface for ontology archiving under consumer-oriented aspects. In: Semantic Systems. In the Era of Knowledge Graphs: 16th International Conference on Semantic Systems, SEMANTiCS 2020, Amsterdam, The Netherlands, September 7–10, 2020, Proceedings 16. pp. 19–35. Springer International Publishing (2020)
17. Gao, L., Golab, L., Özsu, M.T., Aluç, G.: Stream WatDiv: A streaming RDF benchmark. In: Proceedings of the International Workshop on Semantic Big Data. pp. 1–6 (2018)
18. Garijo, D., Corcho, O., Poveda-Villalón, M.: FOOPS!: An ontology pitfall scanner for the FAIR principles. In: ISWC (Posters/Demos/Industry) (2021)
19. Garijo, D., Poveda-Villalón, M.: Best practices for implementing FAIR vocabularies and ontologies on the Web. In: Applications and Practices in Ontology Design, Extraction, and Reasoning, pp. 39–54. IOS Press (2020)
20. Hasemann, H., Kröller, A., Pagel, M.: RDF provisioning for the Internet of Things. In: 2012 3rd IEEE International Conference on the Internet of Things. pp. 143–150. IEEE (2012)
21. Höffner, K., Martin, M., Lehmann, J.: LinkedSpending: OpenSpending becomes Linked Open Data. Semantic Web **7**(1), 95–104 (2016)
22. Käbisch, S., Peintner, D., Anicic, D.: Standardized and efficient RDF encoding for constrained embedded networks. In: European Semantic Web Conference. pp. 437–452. Springer (2015)
23. Karimov, J., Rabl, T., Katsifodimos, A., Samarev, R., Heiskanen, H., Markl, V.: Benchmarking distributed stream data processing systems. In: 2018 IEEE 34th International Conference on Data Engineering (ICDE). pp. 1507–1518. IEEE (2018)
24. Kleppmann, M.: Designing data-intensive applications: The big ideas behind reliable, scalable, and maintainable systems. " O'Reilly Media, Inc." (2017)
25. Kuhn, T., Meroño-Peñuela, A., Malic, A., Poelen, J.H., Hurlbert, A.H., Ortiz, E.C., Furlong, L.I., Queralt-Rosinach, N., Chichester, C., Banda, J.M., et al.: Nanopublications: a growing resource of provenance-centric scientific linked data. In: 2018 IEEE 14th International Conference on e-Science (e-Science). pp. 83–92. IEEE (2018)
26. Le-Phuoc, D., Dao-Tran, M., Pham, M.D., Boncz, P., Eiter, T., Fink, M.: Linked stream data processing engines: Facts and figures. In: The Semantic Web–ISWC 2012: 11th International Semantic Web Conference, Boston, MA, USA, November 11-15, 2012, Proceedings, Part II 11. pp. 300–312. Springer Berlin Heidelberg (2012)
27. Morsey, M., Lehmann, J., Auer, S., Stadler, C., Hellmann, S.: DBpedia and the live extraction of structured data from Wikipedia. Program **46**(2), 157–181 (2012)
28. Nederlands instituut voor Beeld & Geluid: Muziekweb (2023), https://data.muziekweb.nl/, accessed on May 8, 2023
29. Oo, S.M., Haesendonck, G., De Meester, B., Dimou, A.: RMLStreamer-SISO: an RDF stream generator from streaming heterogeneous data. In: The Semantic Web–ISWC 2022: 21st International Semantic Web Conference, Virtual Event, October 23–27, 2022, Proceedings. pp. 697–713. Springer (2022)
30. Patni, H., Henson, C., Sheth, A.: Linked Sensor Data. In: 2010 International Symposium on Collaborative Technologies and Systems. pp. 362–370. IEEE (2010)
31. Pellissier Tanon, T., Weikum, G., Suchanek, F.: YAGO 4: A reason-able knowledge base. In: The Semantic Web: 17th International Conference, ESWC 2020, Heraklion, Crete, Greece, May 31–June 4, 2020, Proceedings 17. pp. 583–596. Springer (2020)

32. Preston-Werner, T.: Semantic versioning 2.0.0 (2023), https://semver.org/, accessed on May 7, 2023
33. RDF Stream Processing Community Group: RSP data model. Draft community group report, W3C RSP Community Group, https://streamreasoning.org/RSP-QL/Abstract%20Syntax%20and%20Semantics%20Document/
34. Schneider, P., Alvarez-Coello, D., Le-Tuan, A., Nguyen-Duc, M., Le-Phuoc, D.: Stream reasoning playground. In: The Semantic Web: 19th International Conference, ESWC 2022, Hersonissos, Crete, Greece, May 29–June 2, 2022, Proceedings. pp. 406–424. Springer (2022)
35. Sowinski, P., Wasielewska-Michniewska, K., Ganzha, M., Pawlowski, W., Szmeja, P., Paprzycki, M.: Efficient RDF streaming for the edge-cloud continuum. arXiv preprint arXiv:2207.04439 (2022), in print, presented on the IEEE 8th World Forum on Internet of Things
36. Szmeja, P., Fornés-Leal, A., Lacalle, I., Palau, C.E., Ganzha, M., Pawłowski, W., Paprzycki, M., Schabbink, J.: ASSIST-IoT: A modular implementation of a reference architecture for the next generation Internet of Things. Electronics **12**(4), 854 (2023)
37. Tommasini, R., Bonte, P.: Web stream processing with RSP4J. In: Proceedings of the 15th ACM International Conference on Distributed and Event-based Systems. pp. 164–167 (2021)
38. Tommasini, R., Bonte, P., Spiga, F., Della Valle, E.: Streaming linked data life cycle. In: Streaming Linked Data: From Vision to Practice, pp. 69–107. Springer (2022)
39. Tommasini, R., Della Valle, E., Mauri, A., Brambilla, M.: RSPLab: RDF stream processing benchmarking made easy. In: The Semantic Web–ISWC 2017: 16th International Semantic Web Conference, Vienna, Austria, October 21-25, 2017, Proceedings, Part II 16. pp. 202–209. Springer (2017)
40. Vandenbussche, P.Y., Atemezing, G.A., Poveda-Villalón, M., Vatant, B.: Linked Open Vocabularies (LOV): a gateway to reusable semantic vocabularies on the Web. Semantic Web **8**(3), 437–452 (2017)
41. Wilkinson, M.D., Dumontier, M., Aalbersberg, I.J., Appleton, G., Axton, M., Baak, A., Blomberg, N., Boiten, J.W., da Silva Santos, L.B., Bourne, P.E., et al.: The FAIR guiding principles for scientific data management and stewardship. Scientific data **3**(1), 1–9 (2016)
42. Zhang, Y., Duc, P.M., Corcho, O., Calbimonte, J.P.: SRBench: a streaming RDF/SPARQL benchmark. In: The Semantic Web–ISWC 2012: 11th International Semantic Web Conference, Boston, MA, USA, November 11-15, 2012, Proceedings, Part I 11. pp. 641–657. Springer (2012)