Jelly-Patch: a Fast Format for Recording Changes in RDF Datasets

Piotr Sowiński [1,2], Kacper Grzymkowski [1], Anastasiya Danilenka [1,2]
[1] NeverBlink, [2] Warsaw University of Technology

Motivation

- **RDF datasets often change** – quads get added and deleted
- **Recording these changes** is useful in:
  - Change data capture
  - Database auditing
  - High-availability RDF databases
  - Event-driven or stream processing systems
- **RDF Patch** allows for that, but the available serialization formats are **too slow** in parsing & serialization, **creating bottlenecks!**

```
# Example RDF Patch file
TX . # Transaction begin
A _:sensor001 <http://ex.org/hasTemperature> "23" . # Add triple
D _:sensor001 <http://ex.org/hasTemperature> "22" . # Delete triple
TC . # Transaction commit
```

Method

New binary serialization format for RDF Patch

- Based on the already proven **Jelly-RDF** binary format
- **Supports quad** add / delete and **prefix** add / delete operations
- A single **Jelly-Patch** file or stream **may contain many patches**
- Uses **Protobuf**, ensuring **portability & easy implementation**
- **Compressed by design** – reduces size of IRIs and repeated terms with a lightweight algorithm

Full transactional support

- **Transaction start, commit, and rollback commands**
- Allows for implementing **distributed transactional RDF DBs**

High-performance Jena & RDF4J implementations

- **Full integration with Apache Jena's RDF Patch stack**
- Low-level integration for **Eclipse RDF4J**
- Reuses **highly-optimized code** from Jelly-JVM for maximum performance
- 100% open-source, Apache 2.0 license

Benchmark datasets

There were no benchmarks for recording RDF changes, so we created our own benchmark datasets, representing **two different, popular use cases**.

Change data capture: `bsbm-cdc`

- Recorded changes from a **Berlin SPARQL Benchmark** run
- **450k patches** (transactions), ~35M triple adds, ~2M triple deletes
- **~9GB** uncompressed in RDF Patch text format

Event streams: `assist-iot-weather`

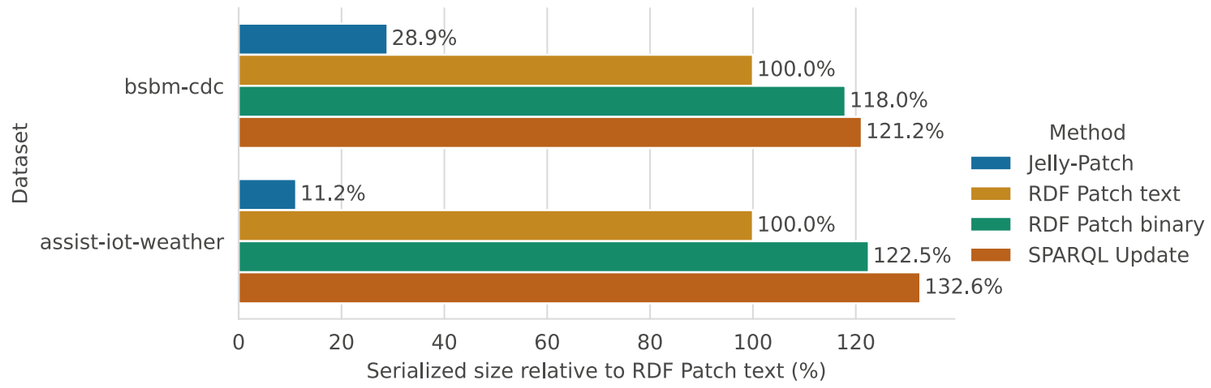
- Rolling diff (delta) of a stream of **IoT weather observations**
- Derived from the **RiverBench** `assist-iot-weather` dataset
- **701k patches** (observations), ~8M triple adds, ~8M triple deletes
- **~2.5GB** uncompressed in RDF Patch text format

Both datasets are available on Zenodo under CC BY 4.0.

Experiments and results

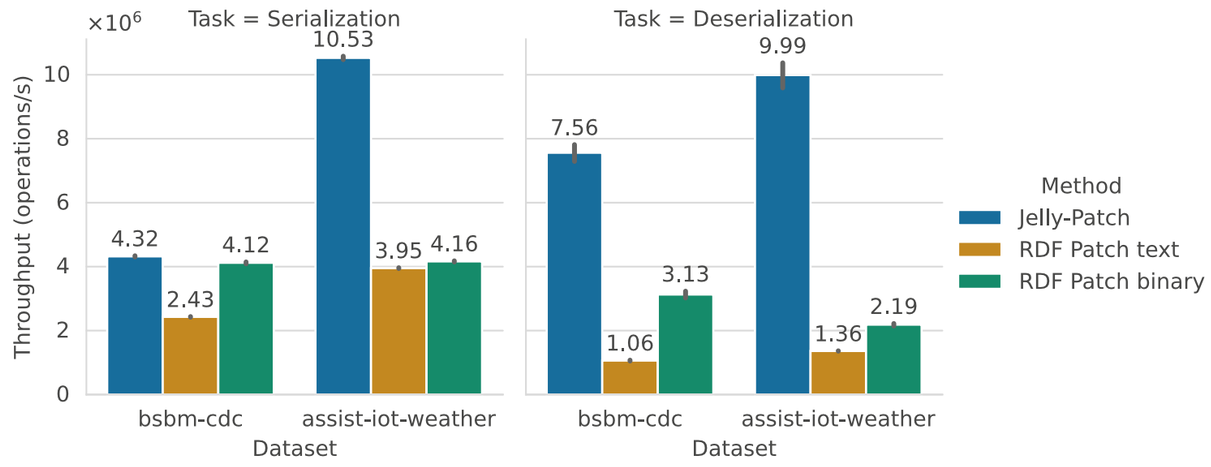
We compared Jelly-Patch with other RDF Patch formats in Apache Jena: RDF Patch text, RDF Patch binary (based on Jena Thrift), and SPARQL Update.

Jelly-Patch **reduced the serialized size** of the patches by **3.5x** for `bsbm-cdc` and **8.9x** for `assist-iot-weather`, as compared to the RDF Patch text format (baseline).



Accidental win – probably the best way to compress IoT data streams?

RDF Patch preserves all information of the original weather data stream, which was **almost 15 GB** in the original (N-Triples). With Jelly-Patch this drops to **279.6 MB (52x size reduction)**. When level 19 zstd compression is applied on top, we achieve **15.3 MB (almost 1000x compression).**



In serialization throughput experiments, Jelly-Patch is **2.5x faster** than any other method in IoT data, and on par with Jena binary in CDC data (pessimistic case for Jelly). In parsing, Jelly-Patch is **4.6x and 2.4x faster**, respectively. We observed speeds up to **10M ops/s** on a single thread.

Conclusion

- The new Jelly-Patch format is **much more compressed & faster** than existing RDF Patch serializations.
- This work answers a direct need in modern, data-intensive RDF systems, **removing a critical performance bottleneck**.
- Jelly-Patch has an **open specification** and **robust, open-source implementations** for Jena and RDF4J.
- **Future work:** investigating uses in RDF stream compression; making a **full, transactional RDF/SPARQL binary protocol**.

We welcome any feedback, feature requests or ideas for how to integrate Jelly-Patch with other software projects!

Paper Datasets Spec

https://w3id.org/jelly