Dydra

&

RDF replication

http://dydra.com

mepdaw 2019

or

transaction-time semantics

≠

valid-time semantics

1. **intent**
2. architecture
3. import time
4. import space
5. query time
6. issues ...

our initial intent was to serve as "github" for data.
that is, to serve as a platform upon which a developer community shared data, built analytical applications upon it, and curated their evolving data over time.

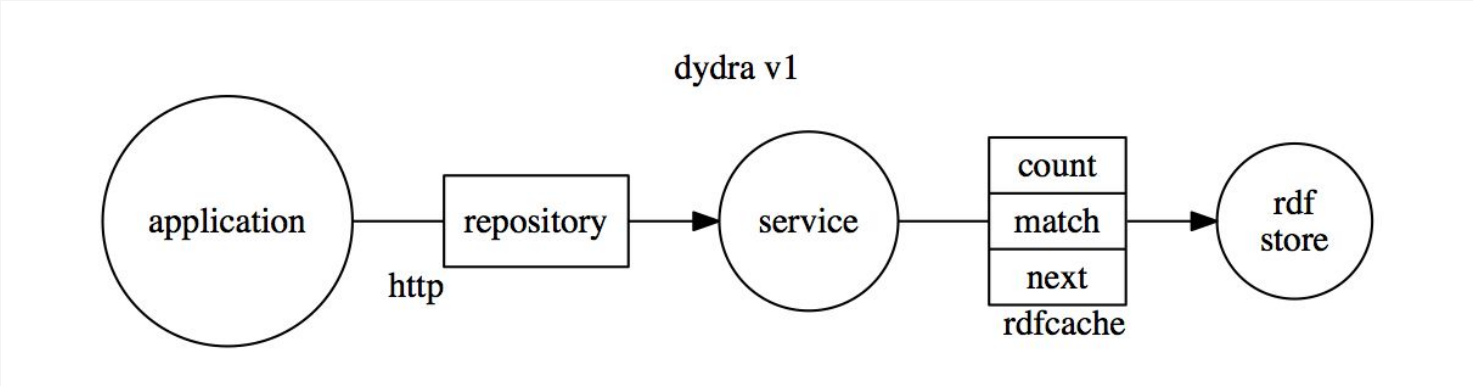
for this use case, an approach based on "independent copies" offered the advantage that deduplication meant innumerable private copies of dbpedia would incur no cost.

on the other hand, we developed early-on the ability to federate to local public repositories at zero-cost, divergent modifications meant some sort of "change-based" approach and use cases which involved frequent updates to production datasets argued for a "timestamp-based" schematic storage model.

over time, the cases have evolved to require not just local static data, but federated access to distributed, dynamic semantic data and the service has evolved to accommodate that, to provide a uniform architecture for

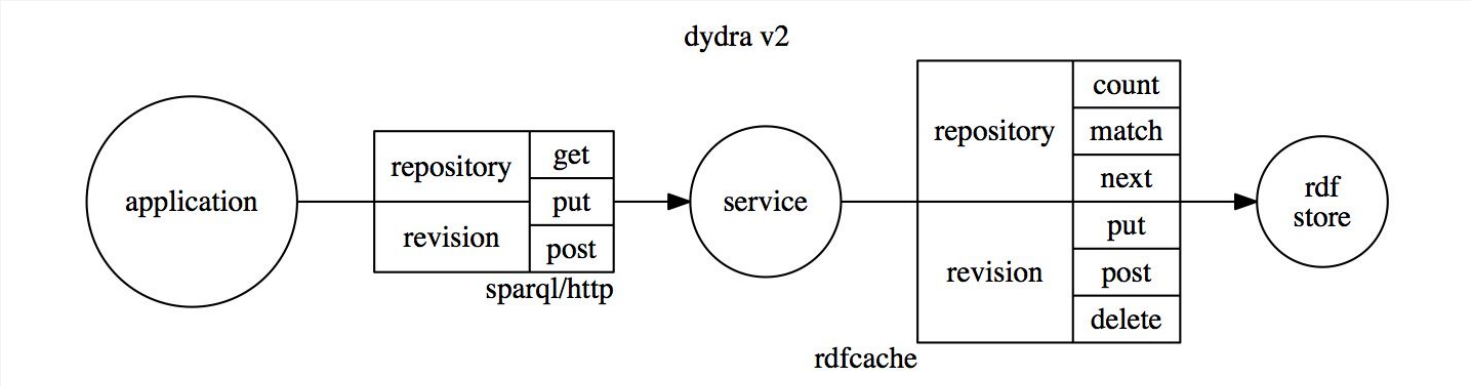
- streaming data
- universal tokenization
- temporally qualified analytics

the initial combination was just sparql:



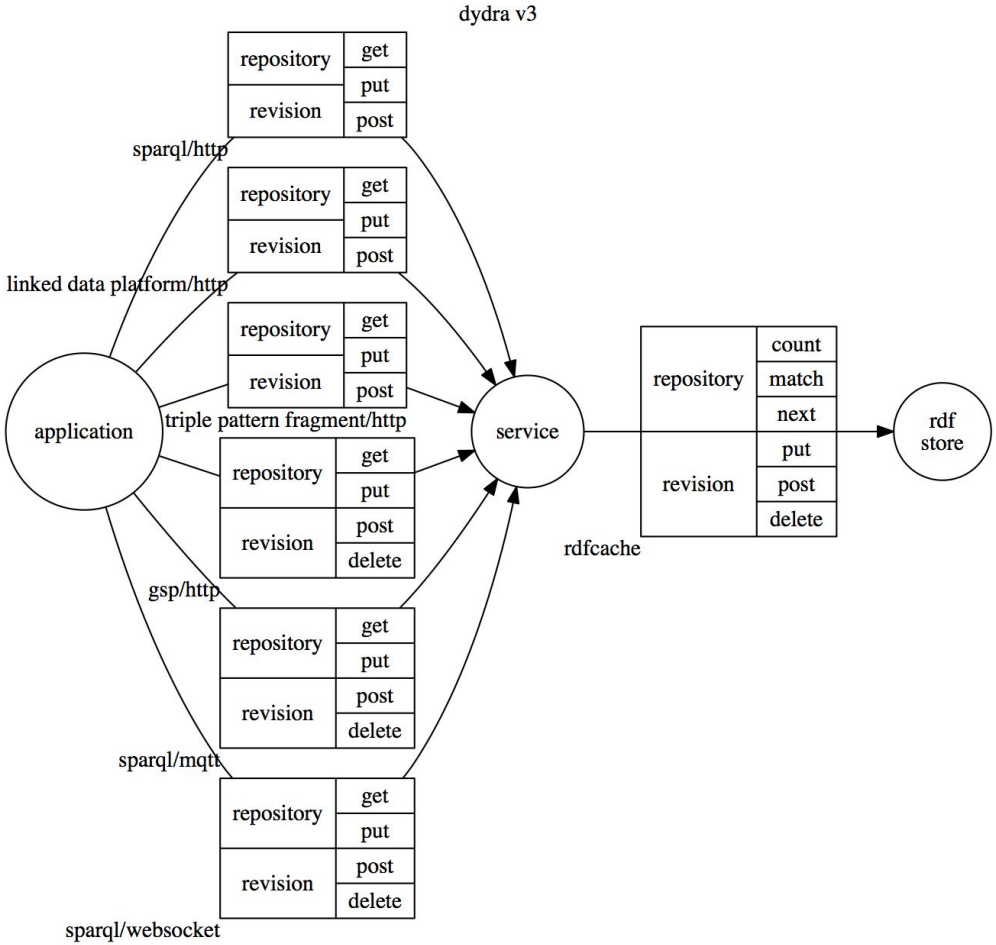
1. intent
2. **architecture**
3. import time
4. import space
5. query time
6. issues …

… this we extended for revisions

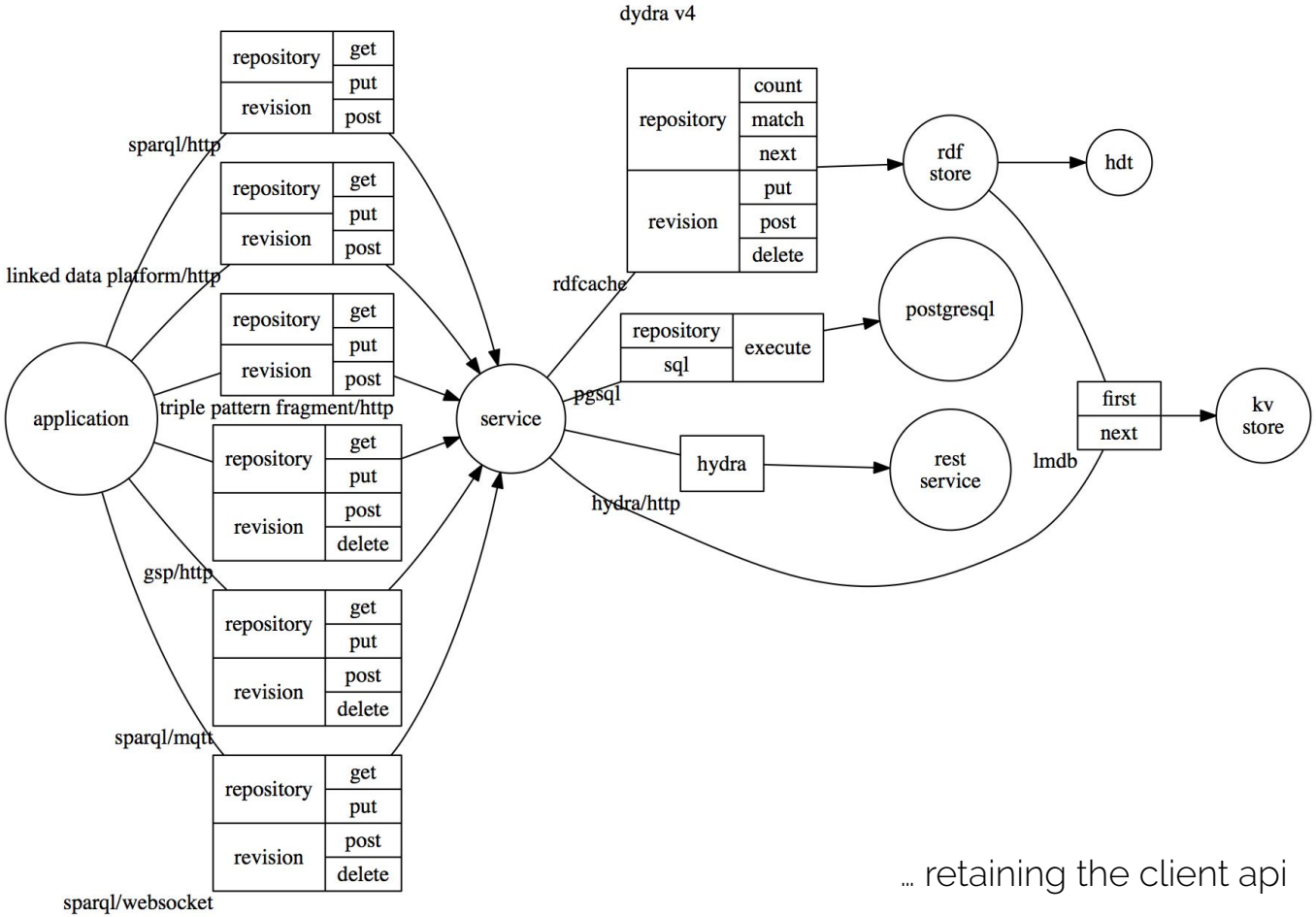


1. intent
2. **architecture**
3. import time
4. import space
5. query time
6. issues …

… and extended the client api to support additional protocols



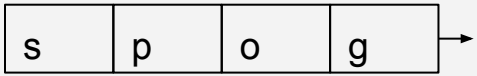
… and extended the back-end to operate on additional sources and on quad index



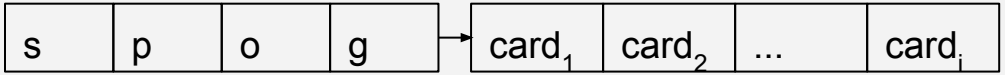
… retaining the client api

alternative index forms

- static index entries :

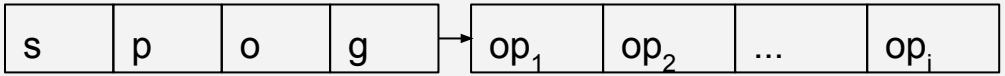


- revisioned index entries :



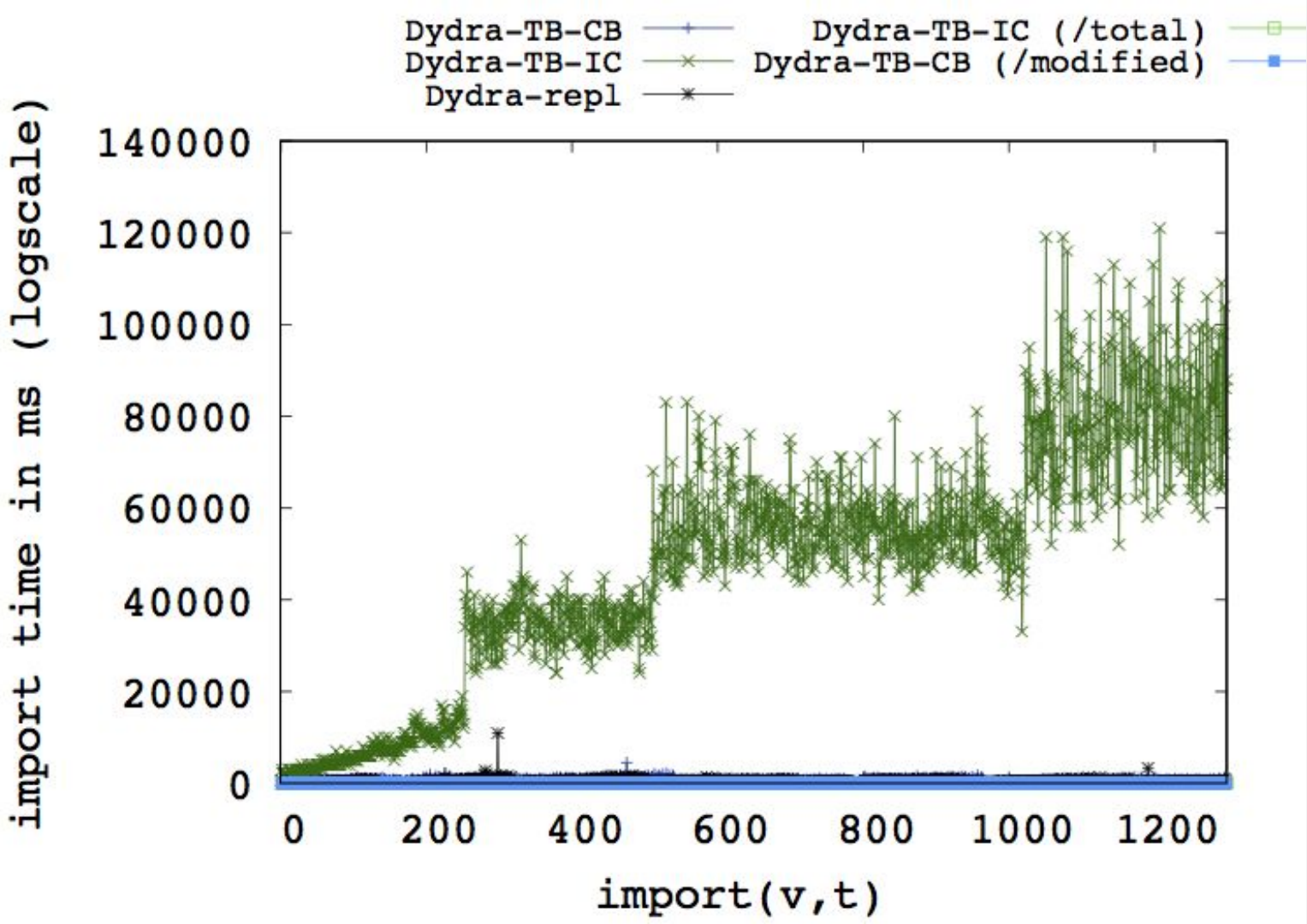
The adoption of timestamps made-up of temporal elements instead of (multi-temporal) simple intervals avoids the duplication of triples in the presence of a temporal pertinence with a complex shape. In fact, we store different triple versions only once with a complex timestamp rather than storing multiple copies of them with a simple timestamp as in [6, 10, 17]. The memory saving we obtain grows with the dimensionality of the time domain, but it can even be appreciated with a monodimensional time domain, when the temporal pertinence of a triple is not a convex interval.[1]

- replicated

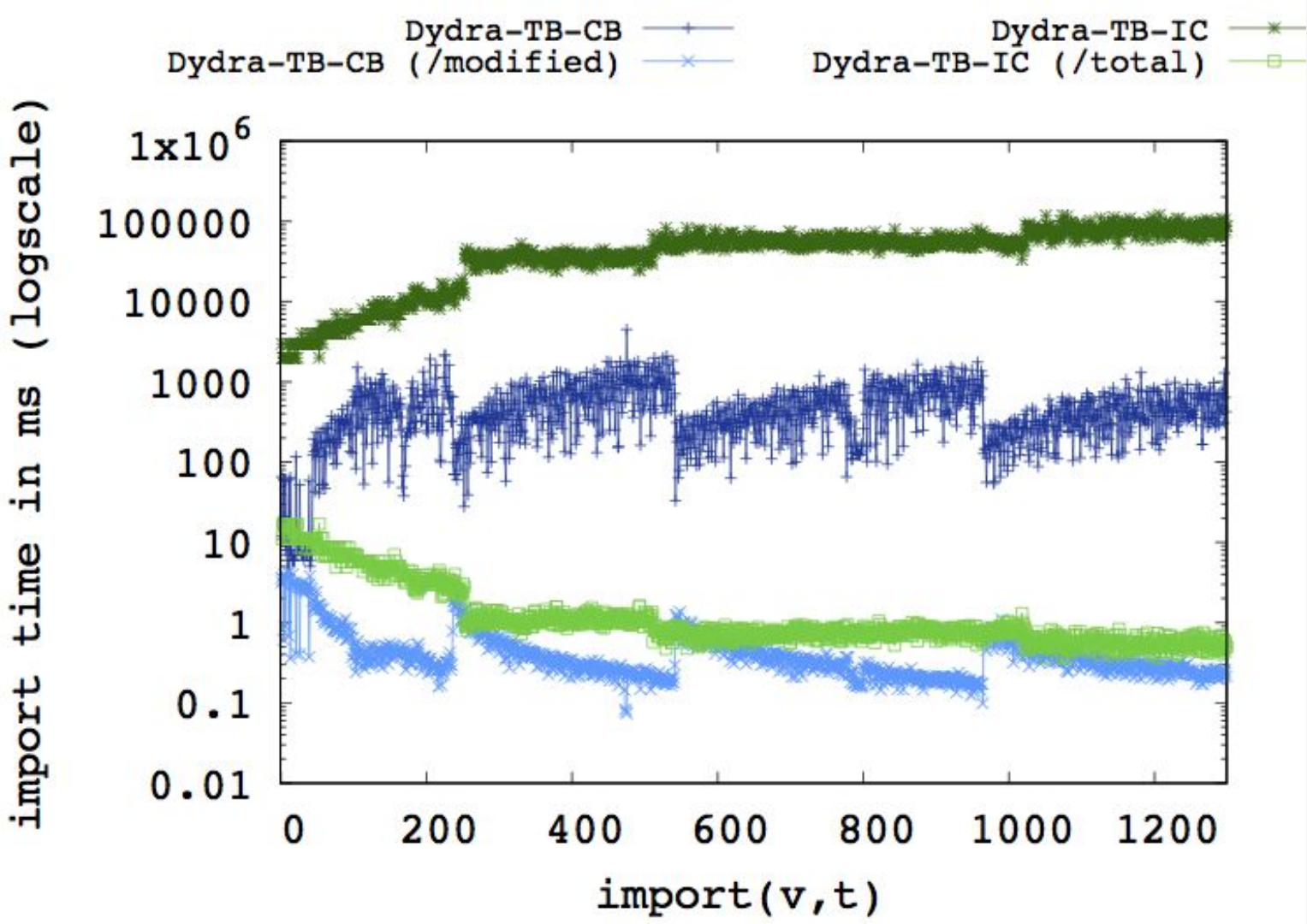


[1]: Grandi.2010 : Grandi F. T-SPARQL: A TSQL2-like Temporal Query Language for RDF

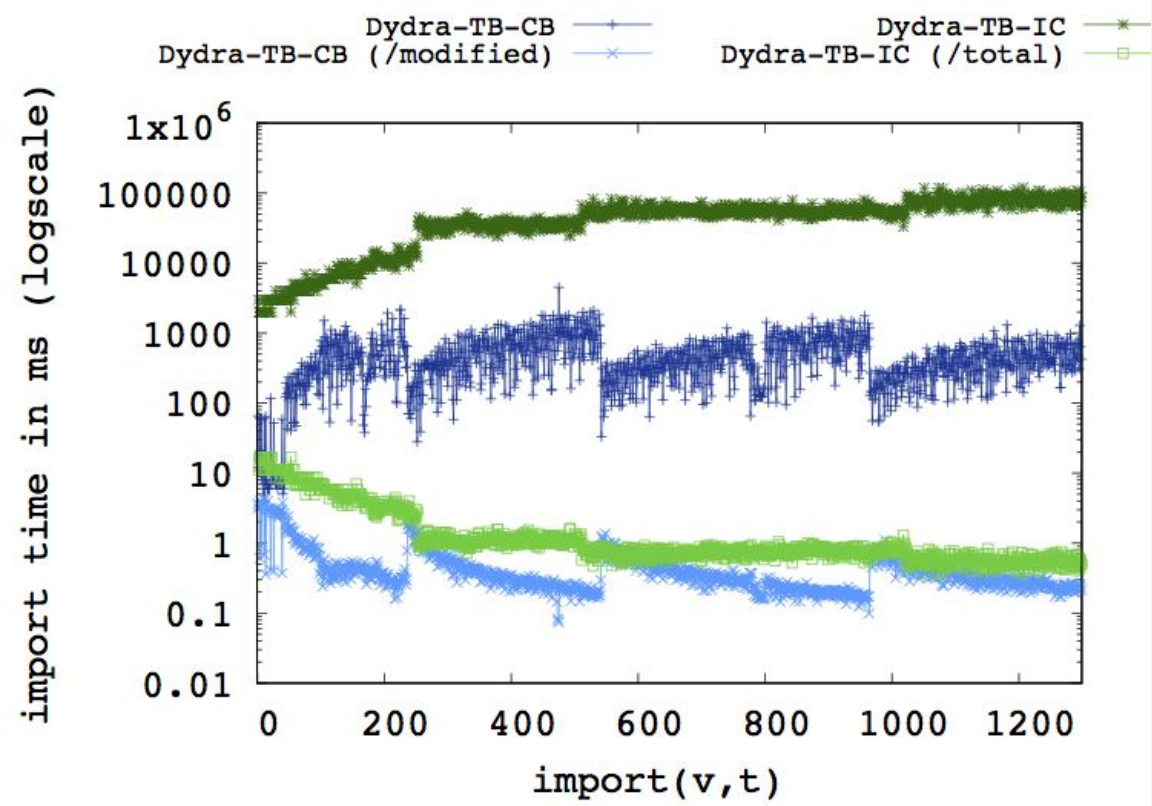
1. intent
2. architecture
3. **import time**
4. import space
5. query time
6. issues ...



1. intent
2. architecture
3. **import time**
4. import space
5. query time
6. issues ...

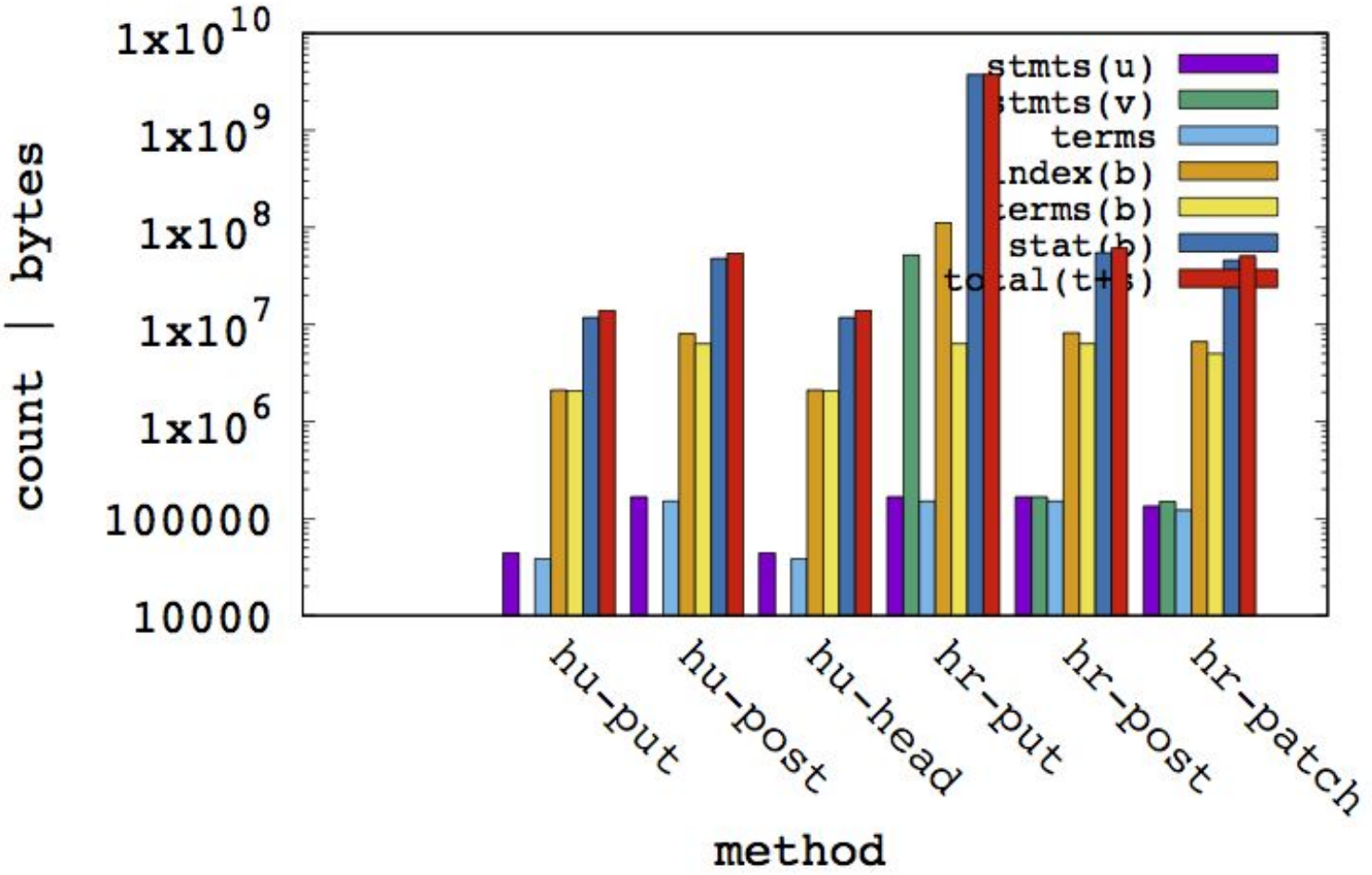


"indepedent copy" (put) v/s "change-based" (patch) import time requirements

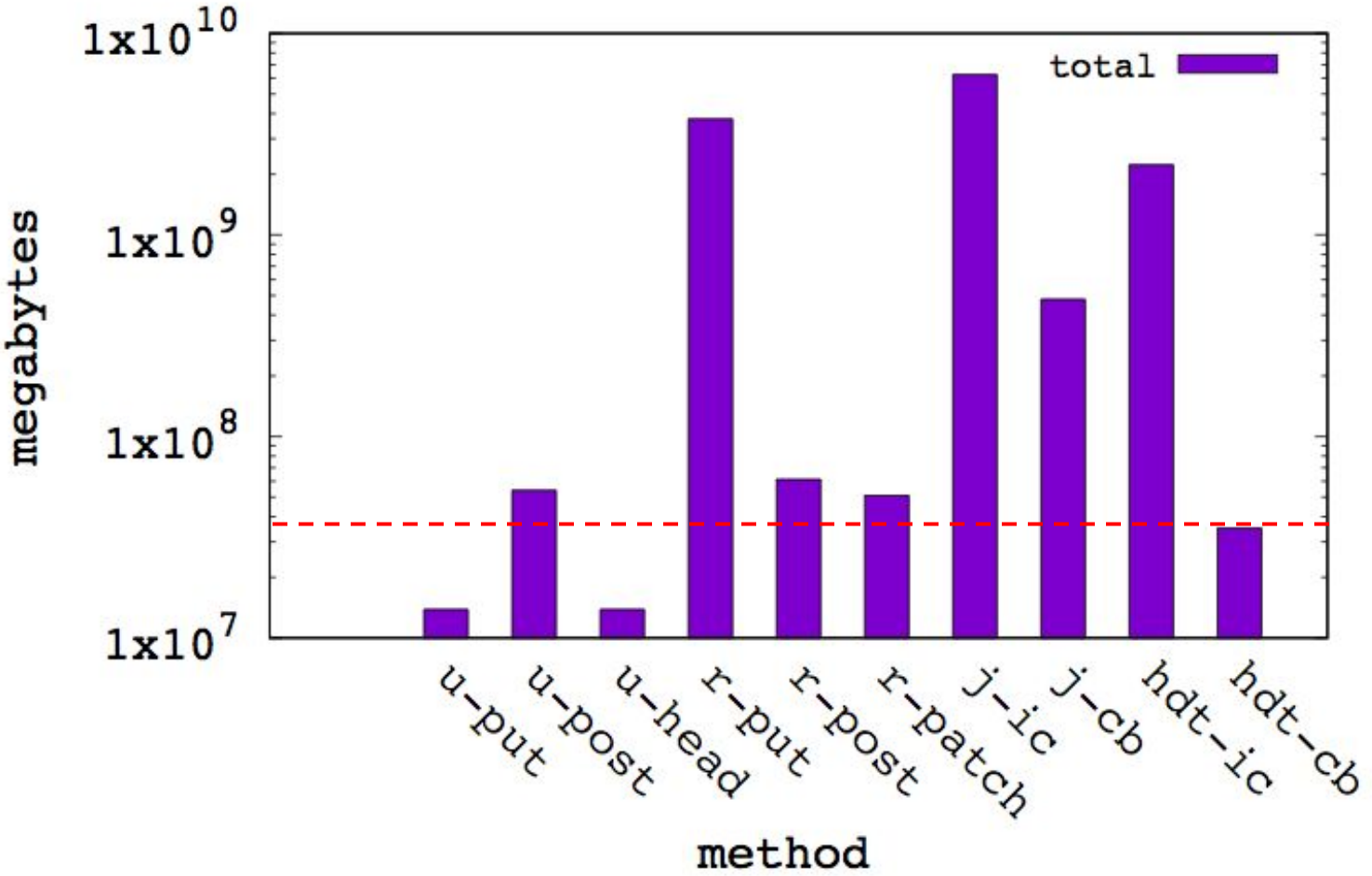


1. intent
2. architecture
3. import time
4. **import space**
5. query time
6. issues ...

bear hour datasets put/post/head space aspects

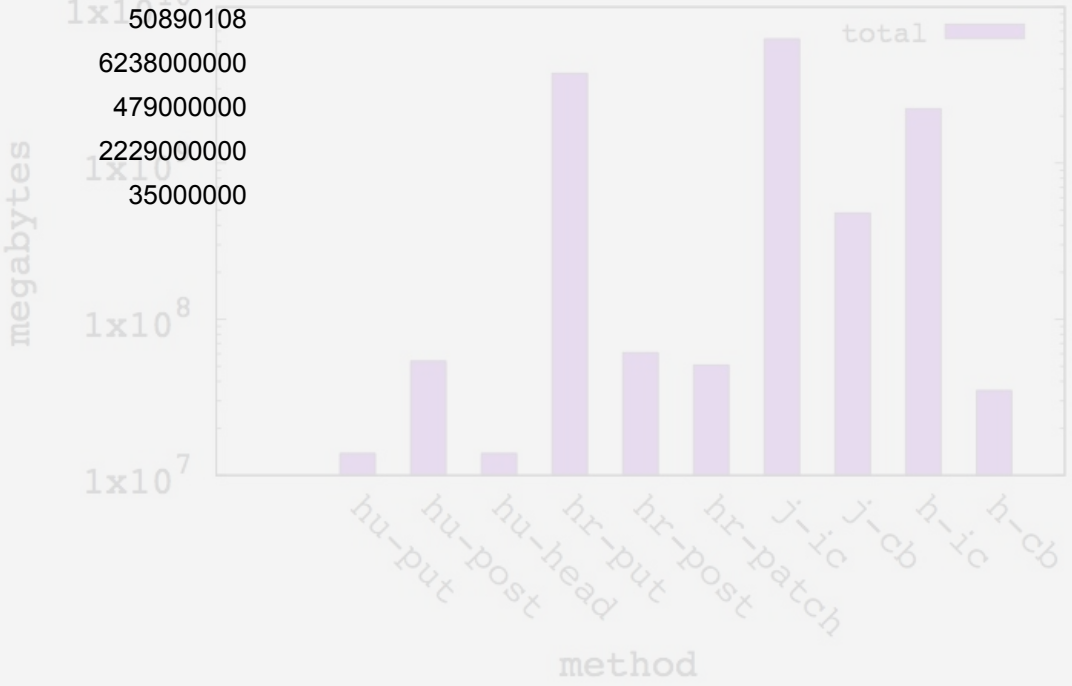


bear hour datasets put/post/head/ic/cb space aspects unrevisioned / revisioned / jena / hdt



"change-based" space requirements :

name	total-bytes
dydra-hu-put	13858594
dydra-hu-post	54008879
dydra-hu-head	13858594
dydra-hr-put	3761601583
dydra-hr-post	61205551
dydra-hr-patch	50890108
jena-tdb-ic	6238000000
jena-tdb-cb	479000000
hdt-ic	2229000000
hdt-cb	35000000



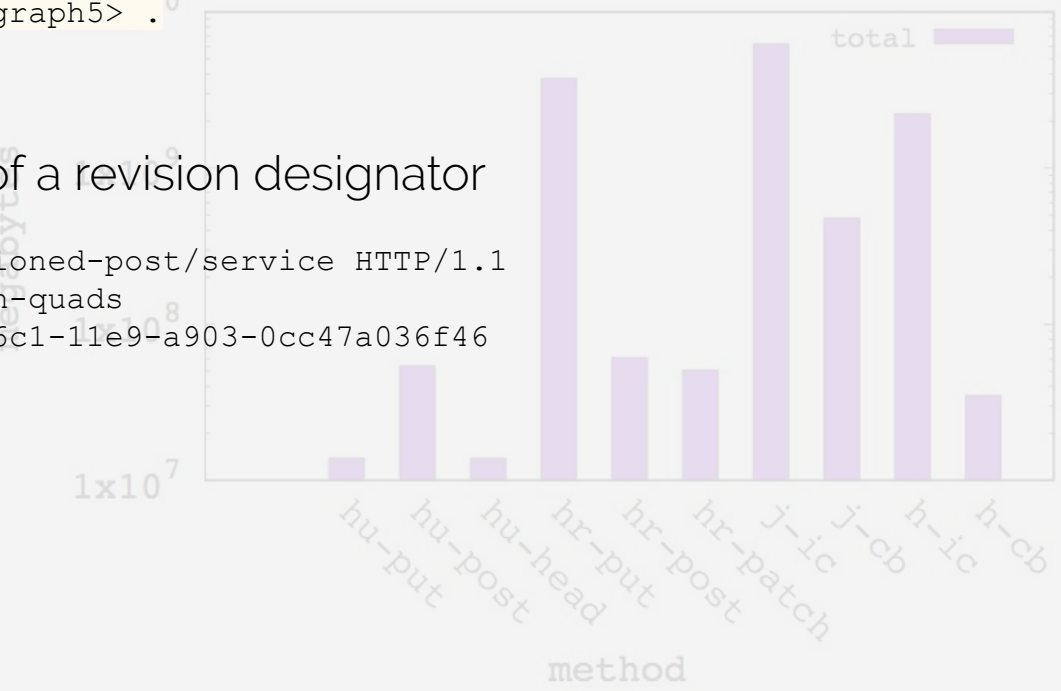
access methods follow the standard gsp interface :

```
POST /bear/hour-revisioned-post/service HTTP/1.1
Content-Type: application/n-quads

<http://ex/s1> <http://ex/p1> <http://ex/o1> <http://example.org/g3> .
 :subject1 <http://an.example/predicate1> "object1"
<http://example.org/graph1> .
 :subject2 <http://an.example/predicate2> "object2"
<http://example.org/graph5> .
...
```

with the addition of a revision designator

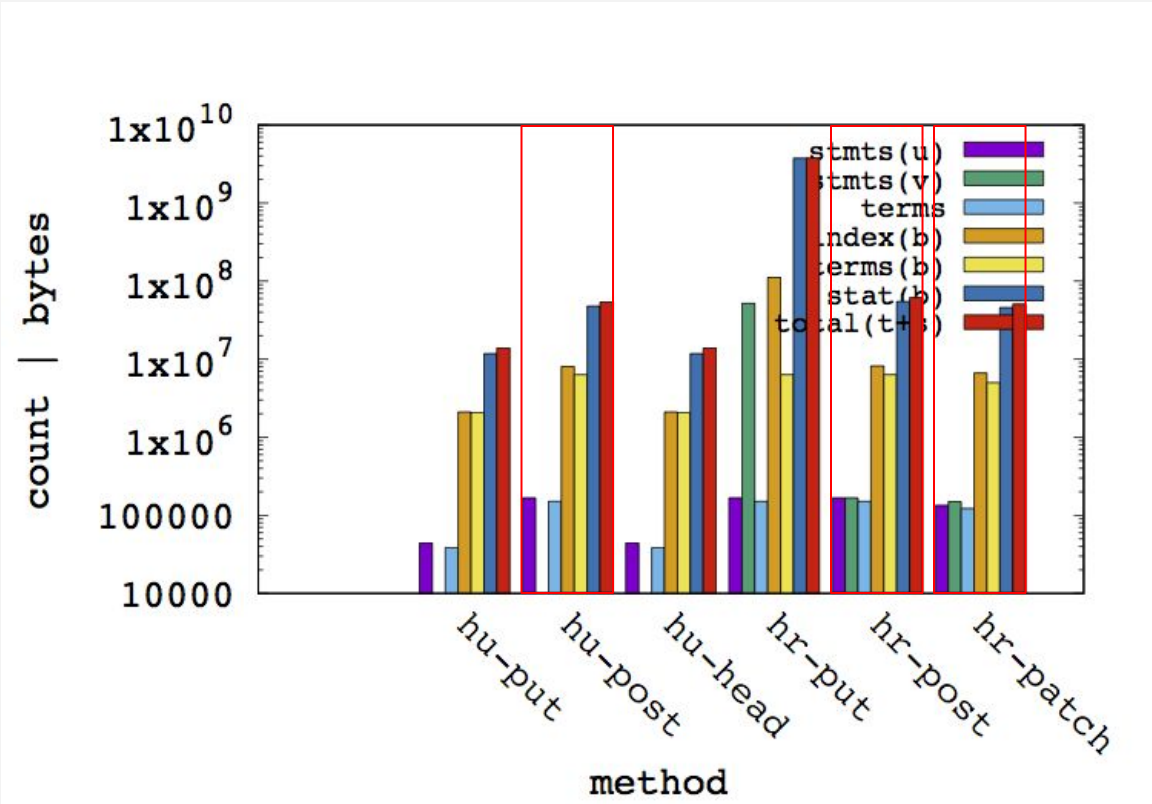
```
GET /bear/hour-revisioned-post/service HTTP/1.1
Accept: application/n-quads
Revision: 382d523a-16c1-11e9-a903-0cc47a036f46
```



1. intent
2. architecture
3. import time
4. **import space**
5. query time
6. issues ...

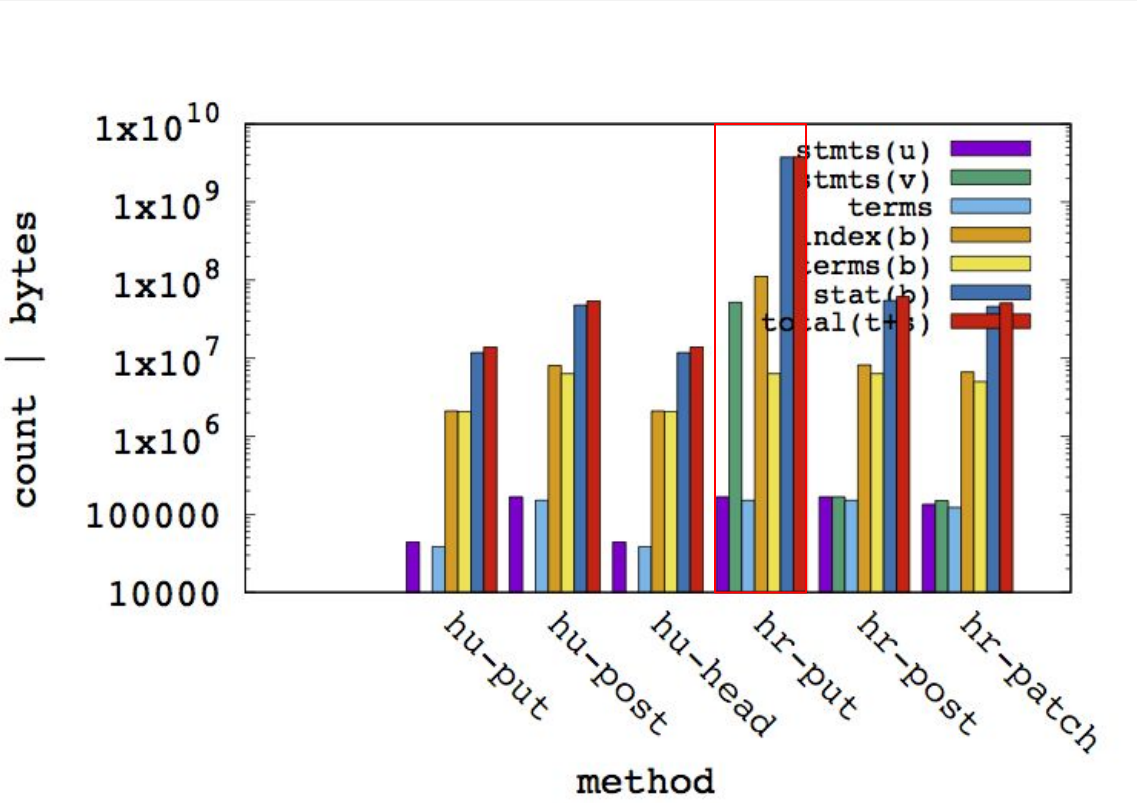
"change-based" space requirements : revisions require little space :

- unrevisioned-post ~= revisioned-post ~= revisioned-patch



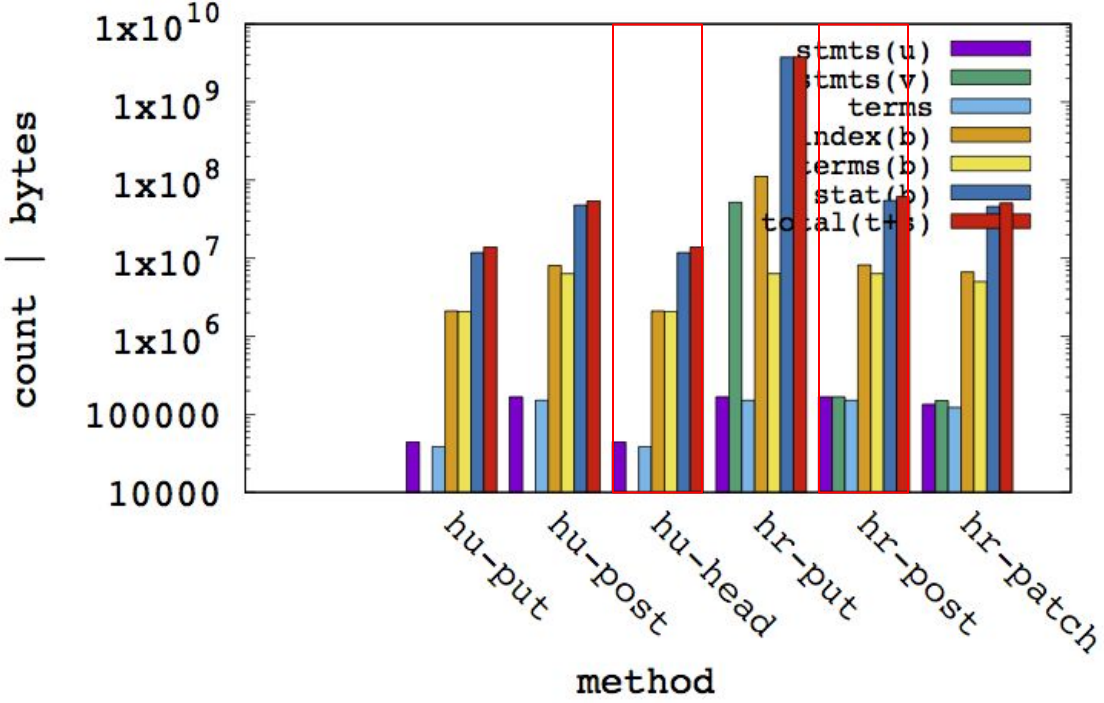
"change-based" space requirements : complete history is costly :

- unrevisioned-post ~= revisioned-post ~= revisioned-patch
- revisioned-put : high churn yields large version map

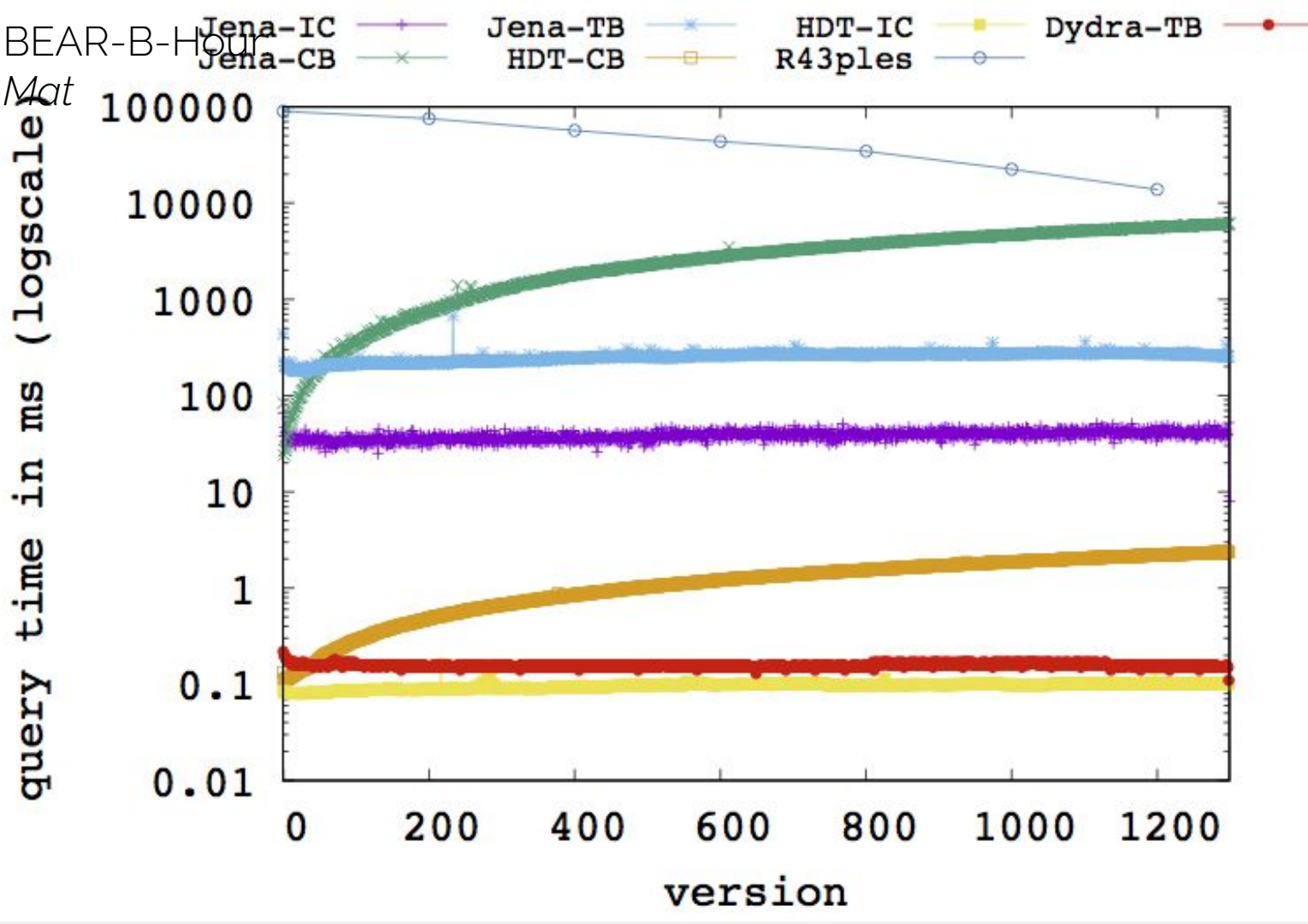


"change-based" space requirements : sublinear in stmt count :

- unrevisioned-post ~= revisioned-post ~= revisioned-patch
- revisioned-put : high churn yields large version map
- (hr-post/hu-head)-bytes / (hr-post/hr-head)-vstmts = 0.80

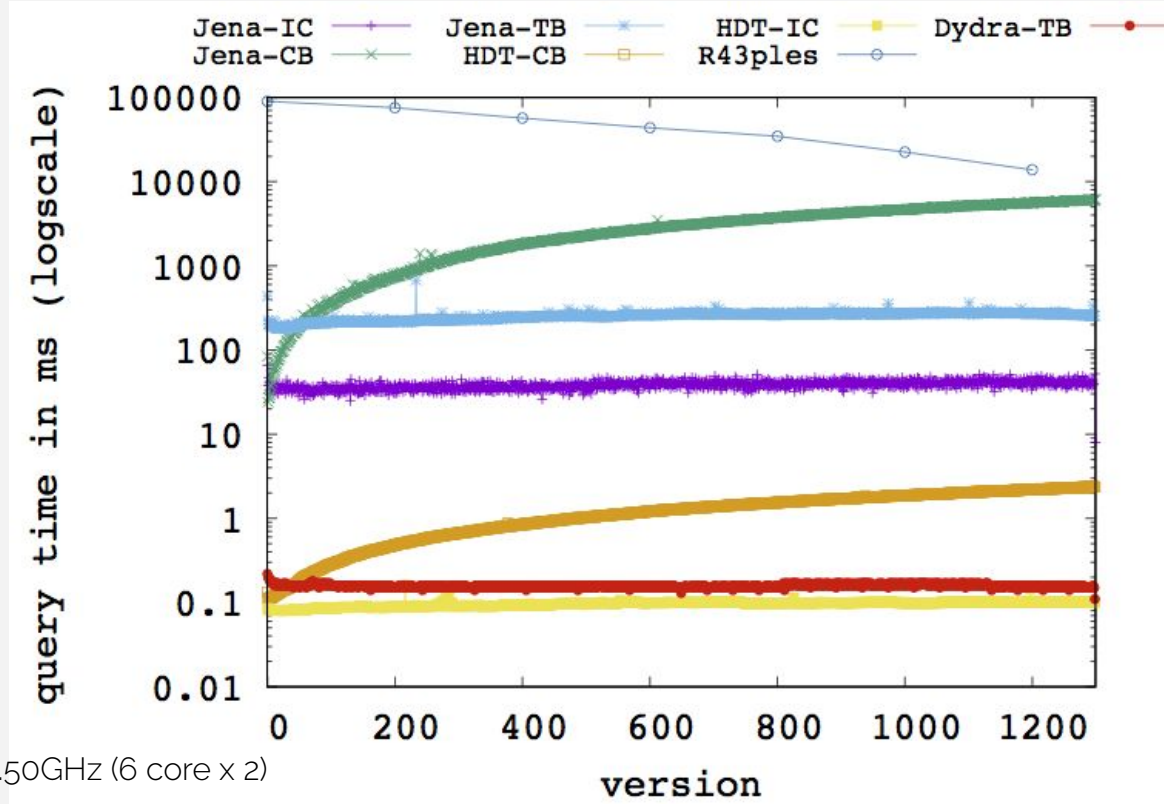


1. intent
2. architecture
3. import time
4. import space
5. **query time**
6. issues …



1. intent
2. architecture
3. import time
4. import space
5. **query time**
6. issues ...

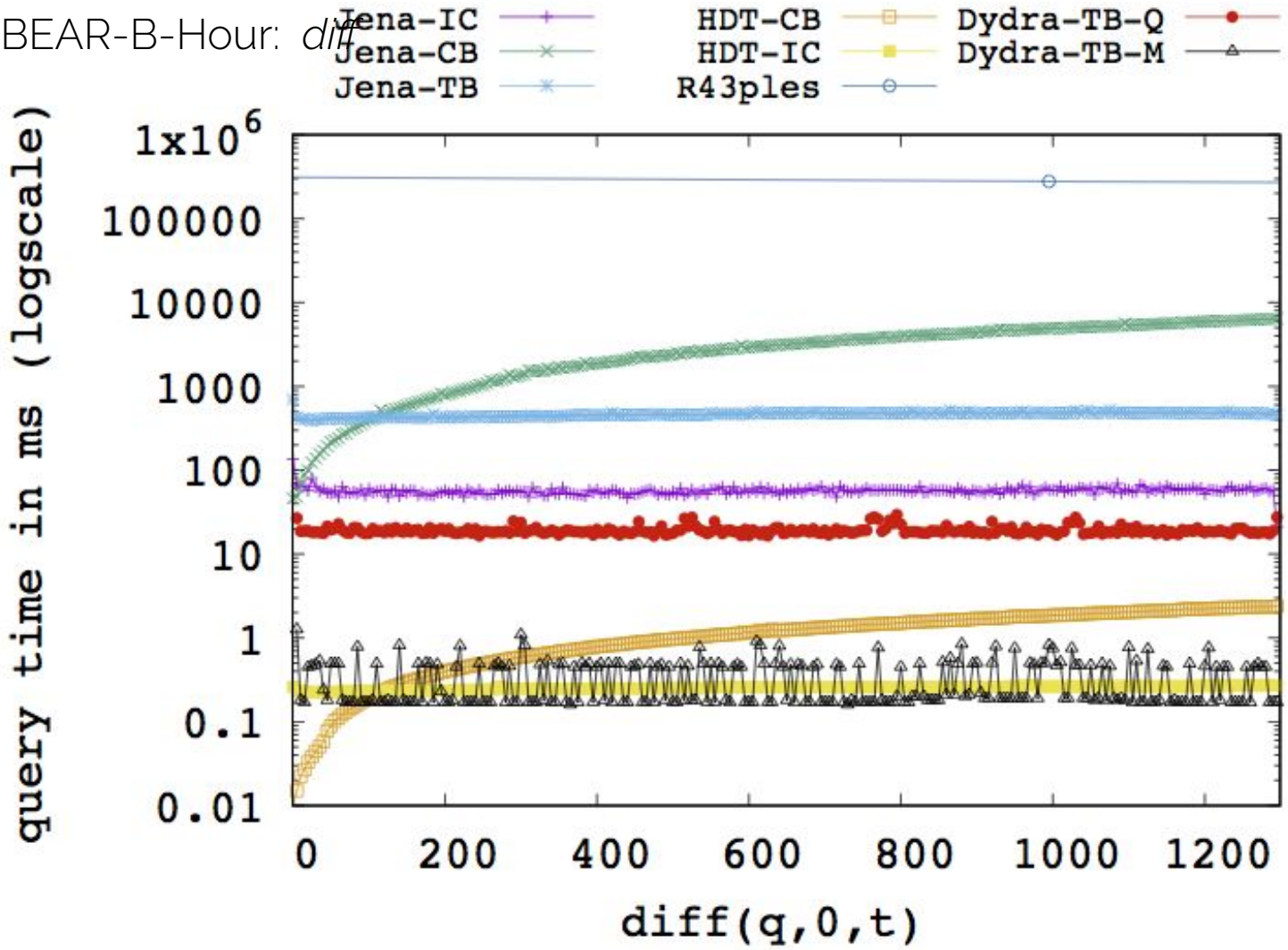
BEAR-B-Hour: *Mat*:
"independent copy" performance



cpu: Intel E5-1650 v3 @ 3.50GHz (6 core x 2)

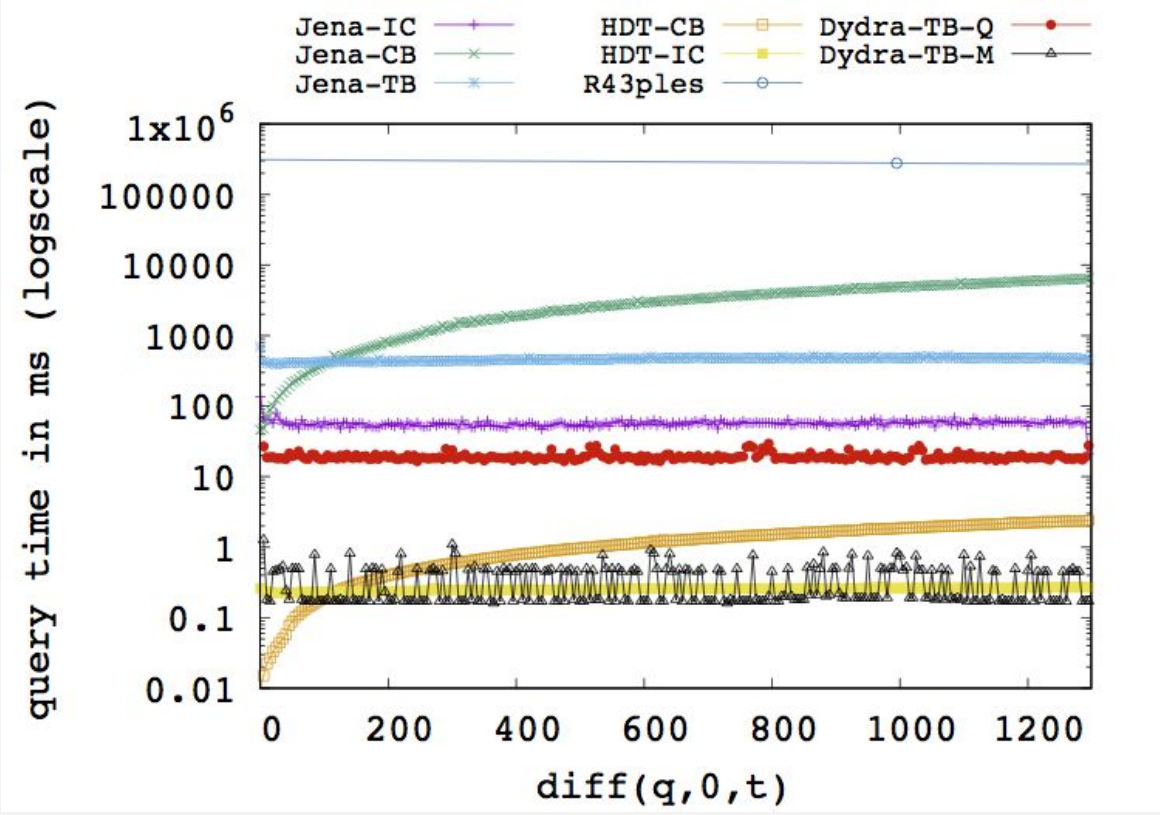
1. intent
2. architecture
3. import time
4. import space
5. **query time**
6. issues …

BEAR-B-Hour: *diff*



1. intent
2. architecture
3. import time
4. import space
5. **query time**
6. issues …

BEAR-B-Hour: *diff*:
"independent copy" performance

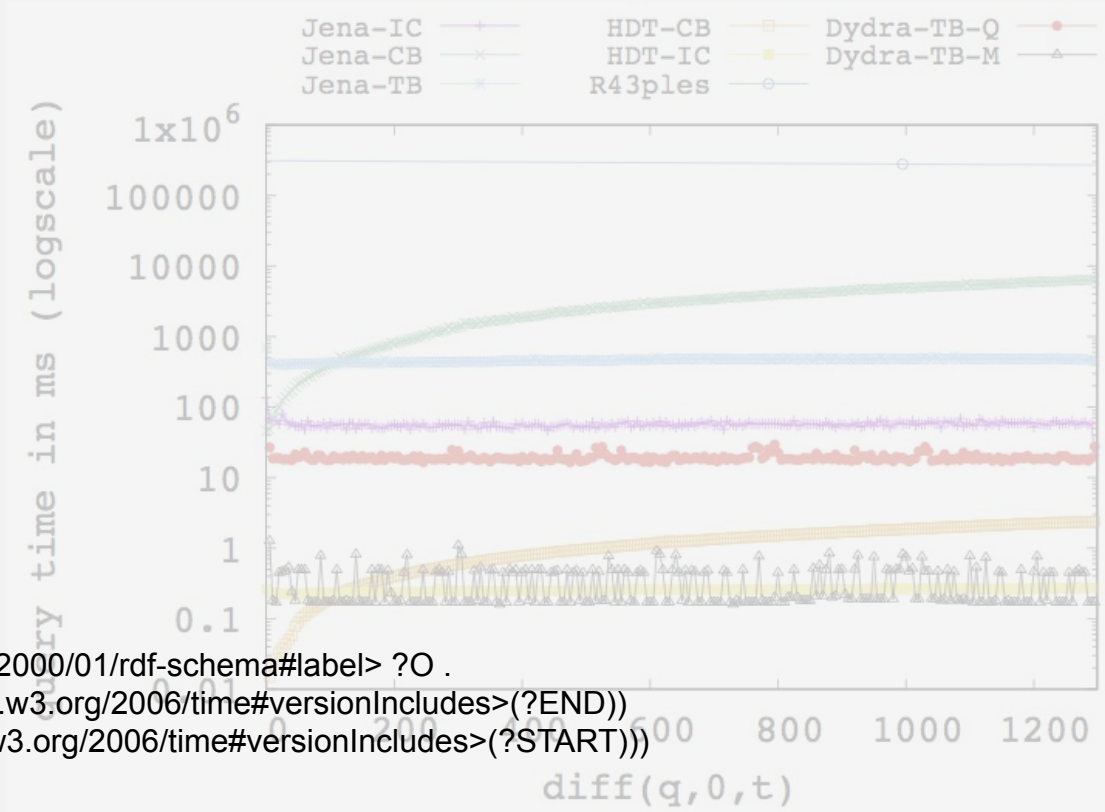


1. intent
2. architecture
3. import time
4. import space
5. **query time**
6. issues ...

BEAR-B-Hour: *diff*:

"independent copy" performance

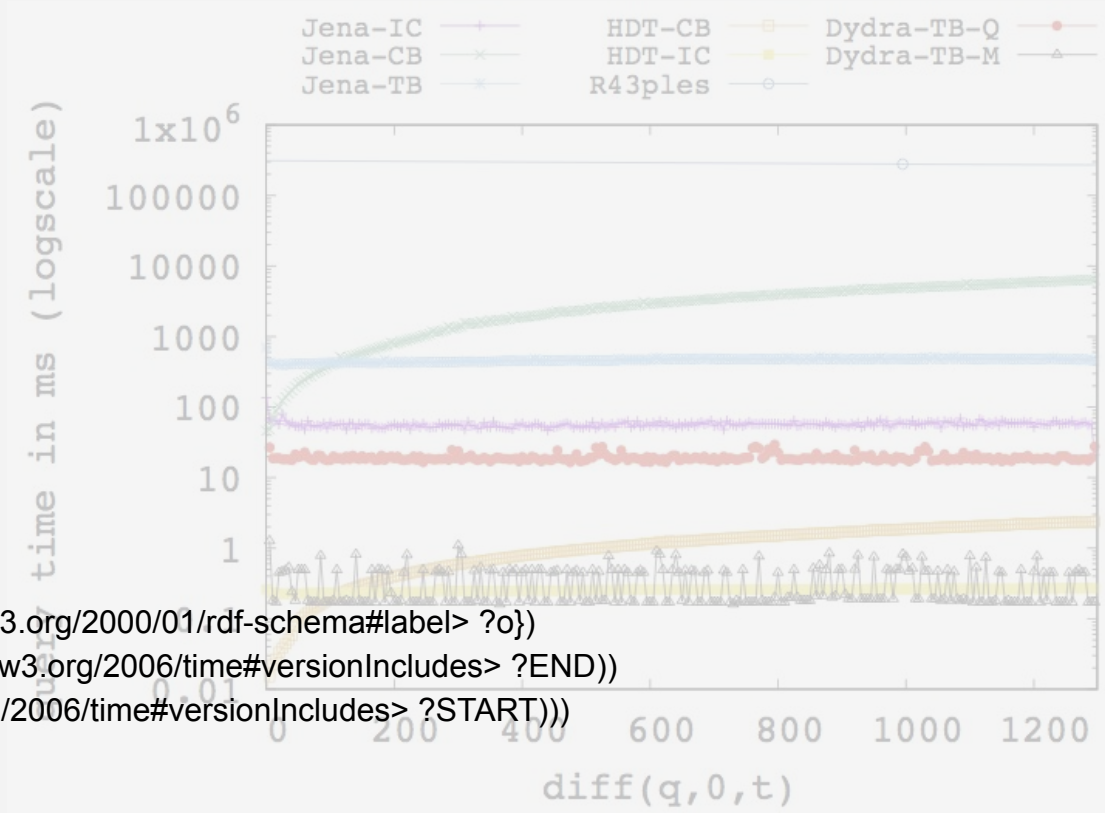
standard SPARQL formulation




```
SELECT ?O ?S
 WHERE {
   ?S <http://www.w3.org/2000/01/rdf-schema#label> ?O .
   FILTER (! ( <http://www.w3.org/2006/time#versionIncludes>(?END))
         && ( <http://www.w3.org/2006/time#versionIncludes>(?START)))
}
```

1. intent
2. architecture
3. import time
4. import space
5. **query time**
6. issues ...

BEAR-B-Hour: *diff*:
"independent copy" performance



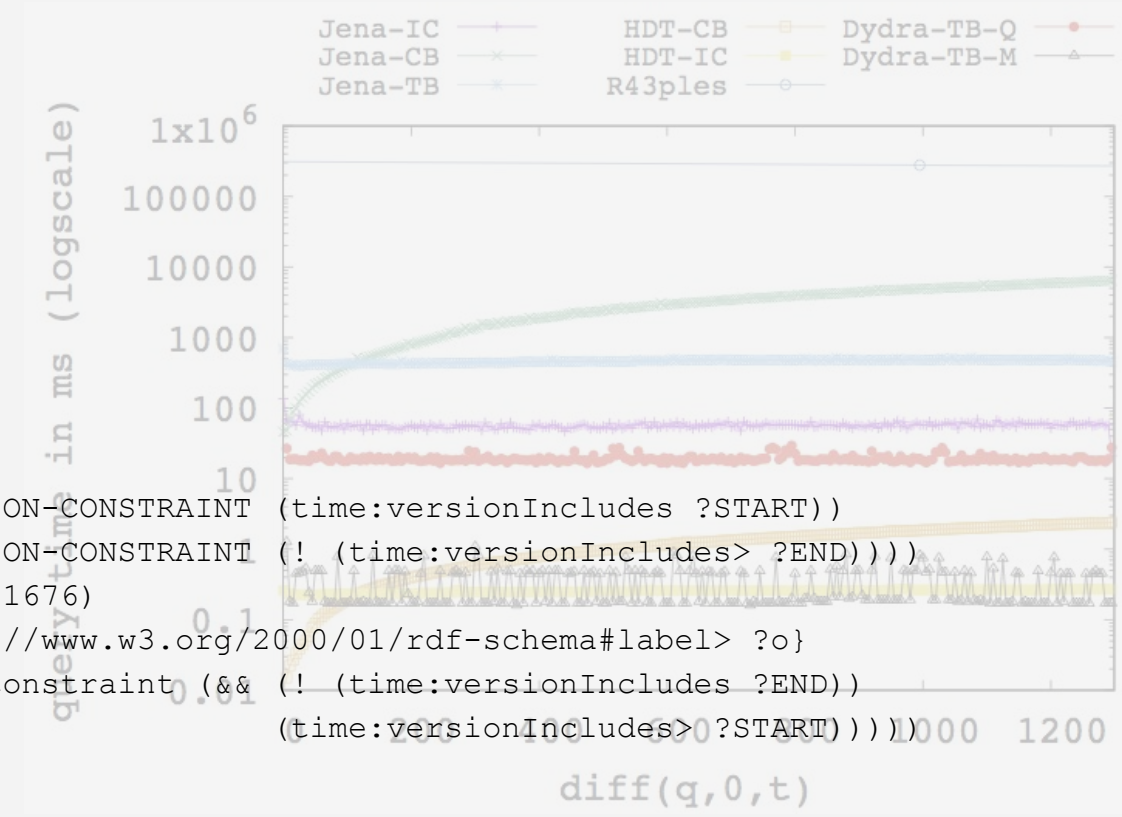

```
(select
  (filter
    (bgp {?s <http://www.w3.org/2000/01/rdf-schema#label> ?o})
    (and (not (<http://www.w3.org/2006/time#versionIncludes> ?END))
         (<http://www.w3.org/2006/time#versionIncludes> ?START)))
  (?o ?s))
```

1. intent
2. architecture
3. import time
4. import space
5. **query time**
6. issues …

BEAR-B-Hour: *diff*:

"independent copy" performance

compiled to exploit temporal index

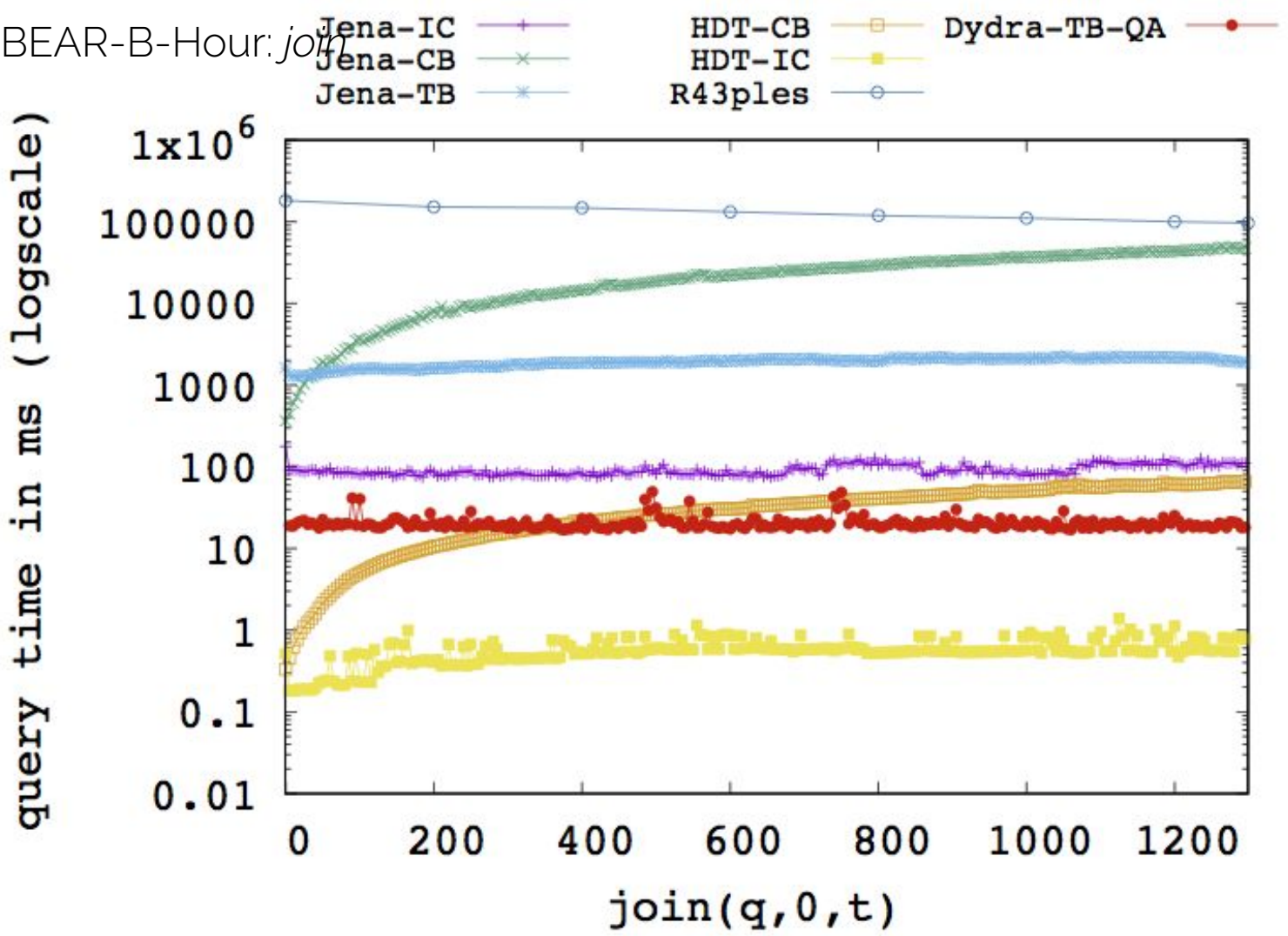



```
(PROJECT
 (LOCALLY
   (DECLARE (VERSION-CONSTRAINT (time:versionIncludes ?START))
            (VERSION-CONSTRAINT (! (time:versionIncludes> ?END))))
   (agp (id #:AGP-1676)
        {?s <http://www.w3.org/2000/01/rdf-schema#label> ?o}
        (version-constraint (&& (! (time:versionIncludes ?END))
                                (time:versionIncludes> ?START)))))

 '(?o ?s))
```

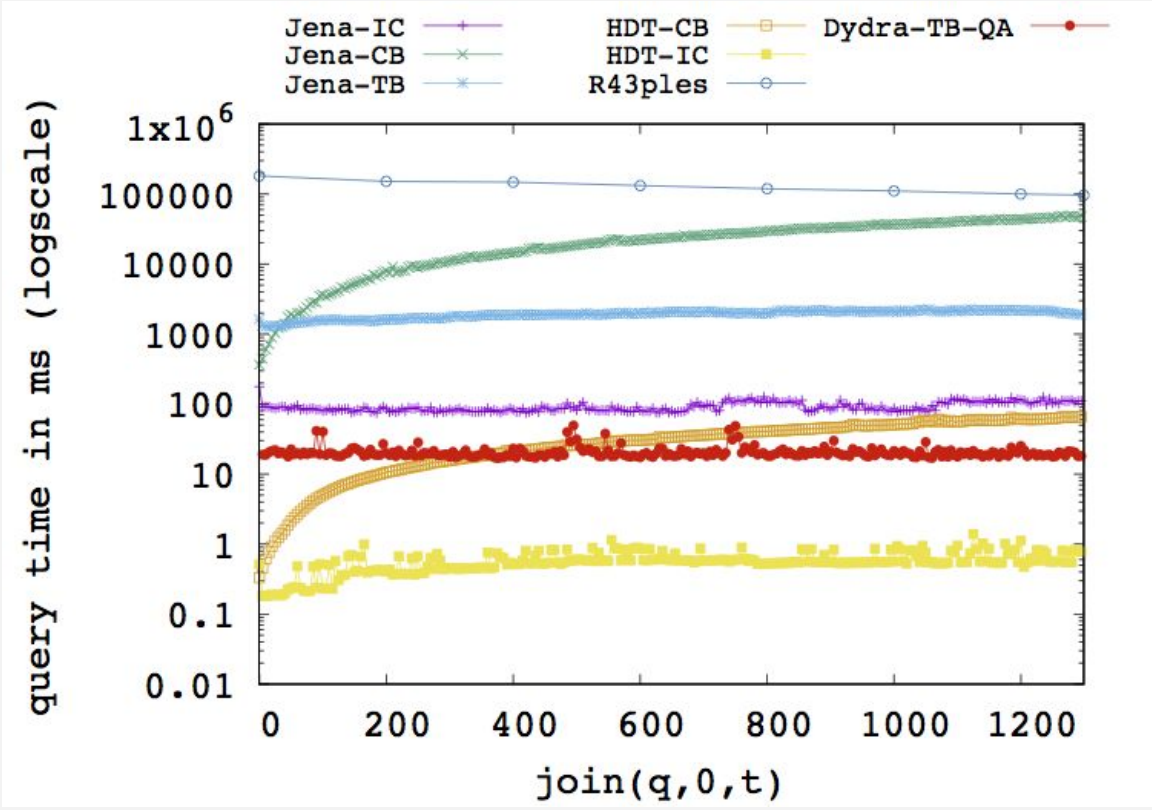
1. intent
2. architecture
3. import time
4. import space
5. **query time**
6. issues …

BEAR-B-Hour: *join*



1. intent
2. architecture
3. import time
4. import space
5. **query time**
6. issues ...

BEAR-B-Hour: *join* :
"independent copy" performance

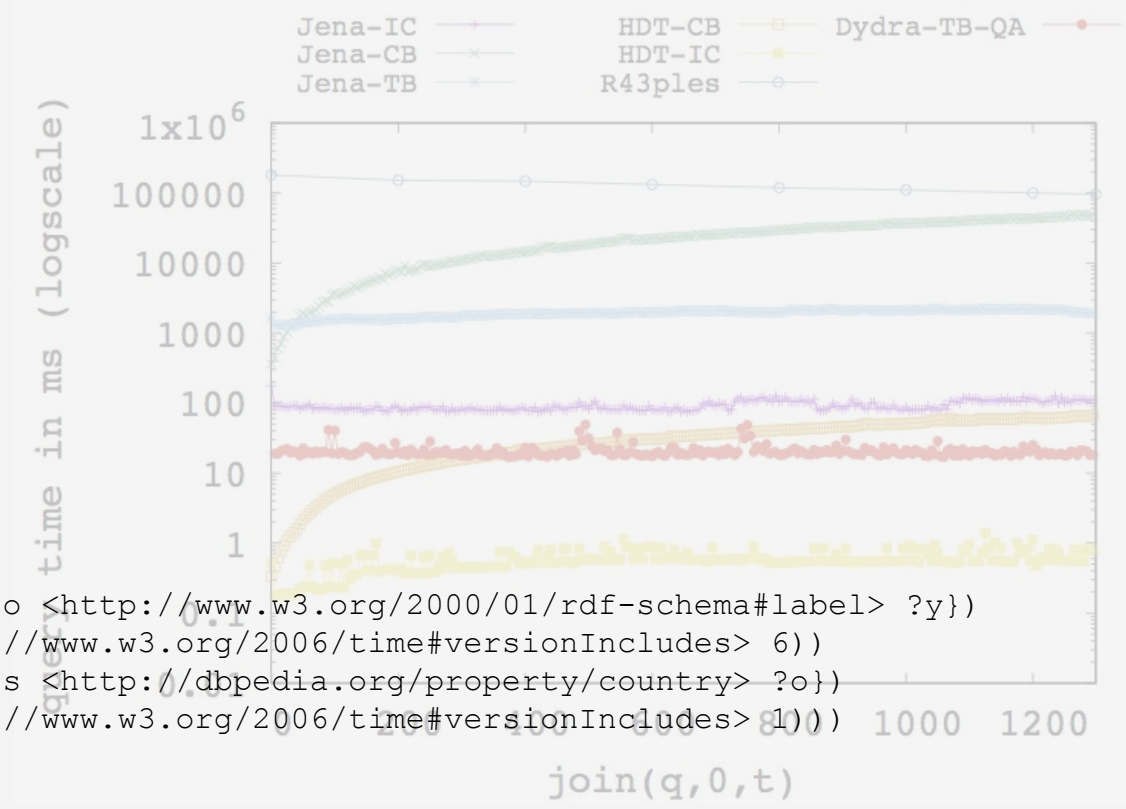


1. intent
2. architecture
3. import time
4. import space
5. **query time**
6. issues ...

temporal constraints are expressed as filters with a lexical scope which covers the intended graph.


```
SELECT ?o ?s ?y
WHERE {
  { ?s <http://dbpedia.org/property/country> ?o
    FILTER ( <http://www.w3.org/2006/time#versionIncludes>(1)) }
  { ?o <http://www.w3.org/2000/01/rdf-schema#label> ?y
     FILTER ( <http://www.w3.org/2006/time#versionIncludes>(6)) }
}
```

this relation is captured in the sse combinations, ...

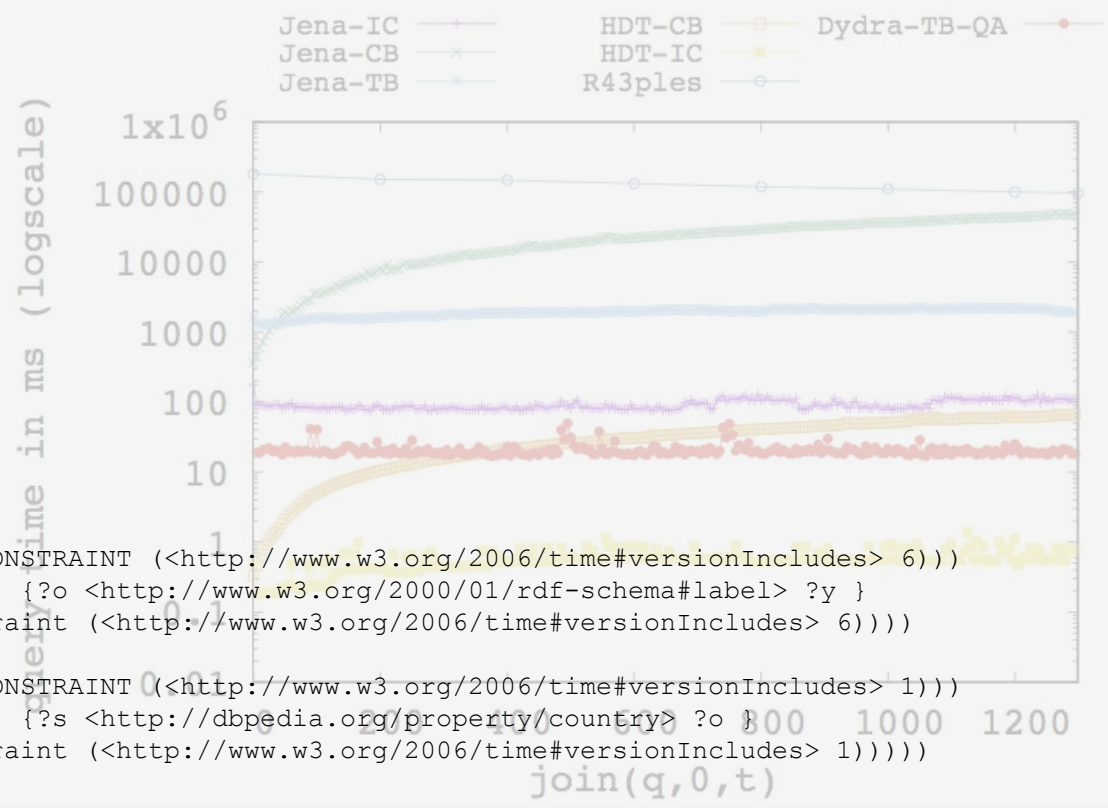



```
(select
  (join
    (filter (bgp {?o <http://www.w3.org/2000/01/rdf-schema#label> ?y})
            (<http://www.w3.org/2006/time#versionIncludes> 6))
    (filter (bgp {?s <http://dbpedia.org/property/country> ?o})
            (<http://www.w3.org/2006/time#versionIncludes> 1)))
 (?o ?s ?y))
```

1. intent
2. architecture
3. import time
4. import space
5. **query time**
6. issues …

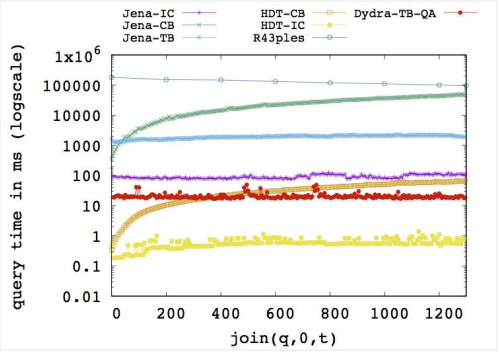
… which are recognized by the compiler and transformed into constraints into version constraints to be applied during bgp processing

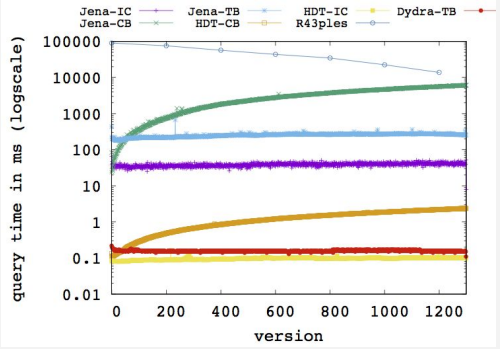
…

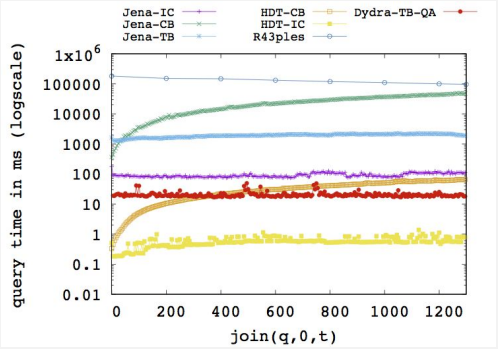



```
(PROJECT
 (JOIN
  (LOCALLY
    (DECLARE (VERSION-CONSTRAINT (<http://www.w3.org/2006/time#versionIncludes> 6)))
    (agp (id #:AGP-1702) {?o <http://www.w3.org/2000/01/rdf-schema#label> ?y }
         (version-constraint (<http://www.w3.org/2006/time#versionIncludes> 6))))
  (LOCALLY
    (DECLARE (VERSION-CONSTRAINT (<http://www.w3.org/2006/time#versionIncludes> 1)))
    (agp (id #:AGP-1704) {?s <http://dbpedia.org/property/country> ?o }
         (version-constraint (<http://www.w3.org/2006/time#versionIncludes> 1)))))
'(?o ?s ?y))
```

1. intent
2. architecture
3. import time
4. import space
5. query time
6. issues …

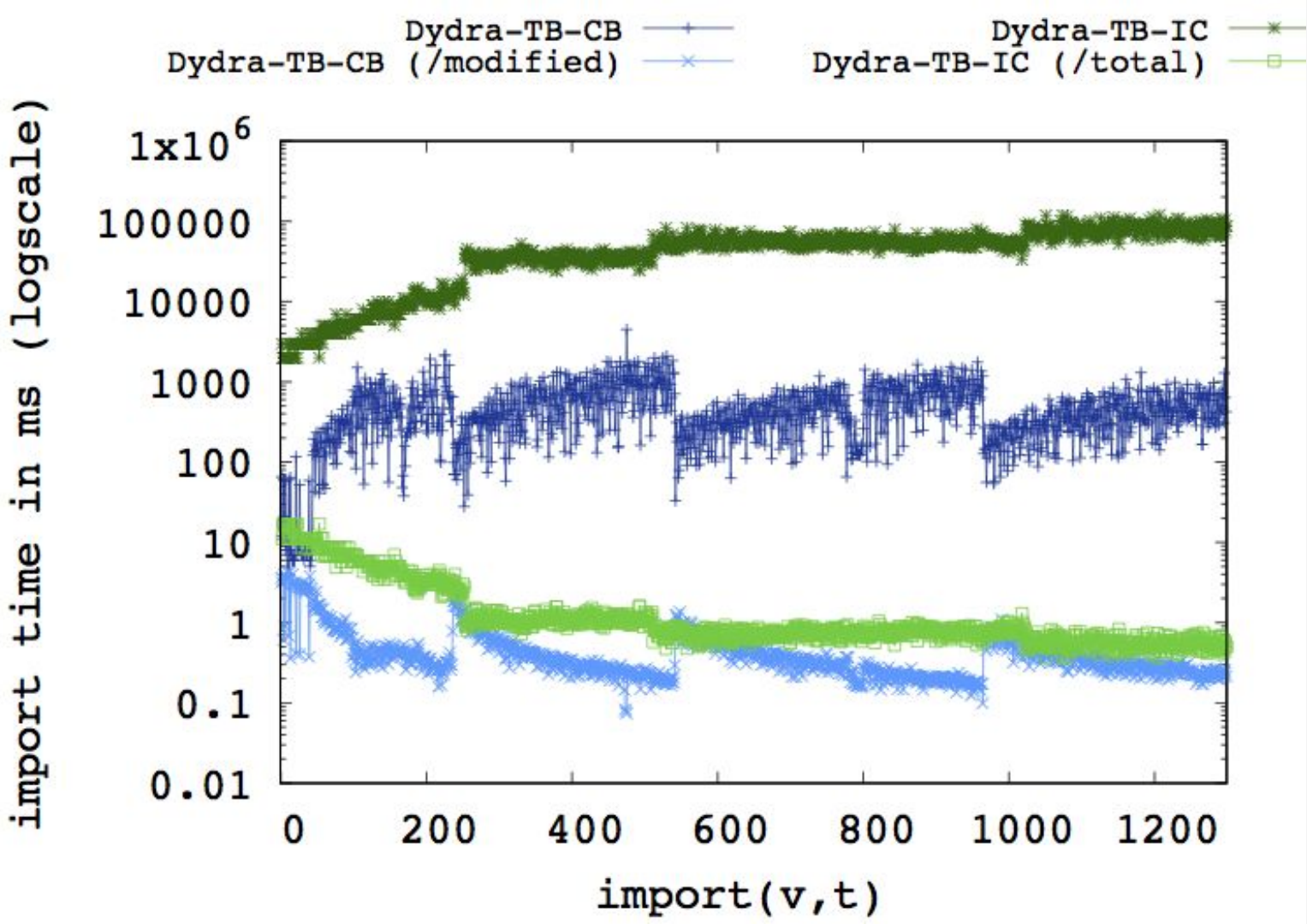






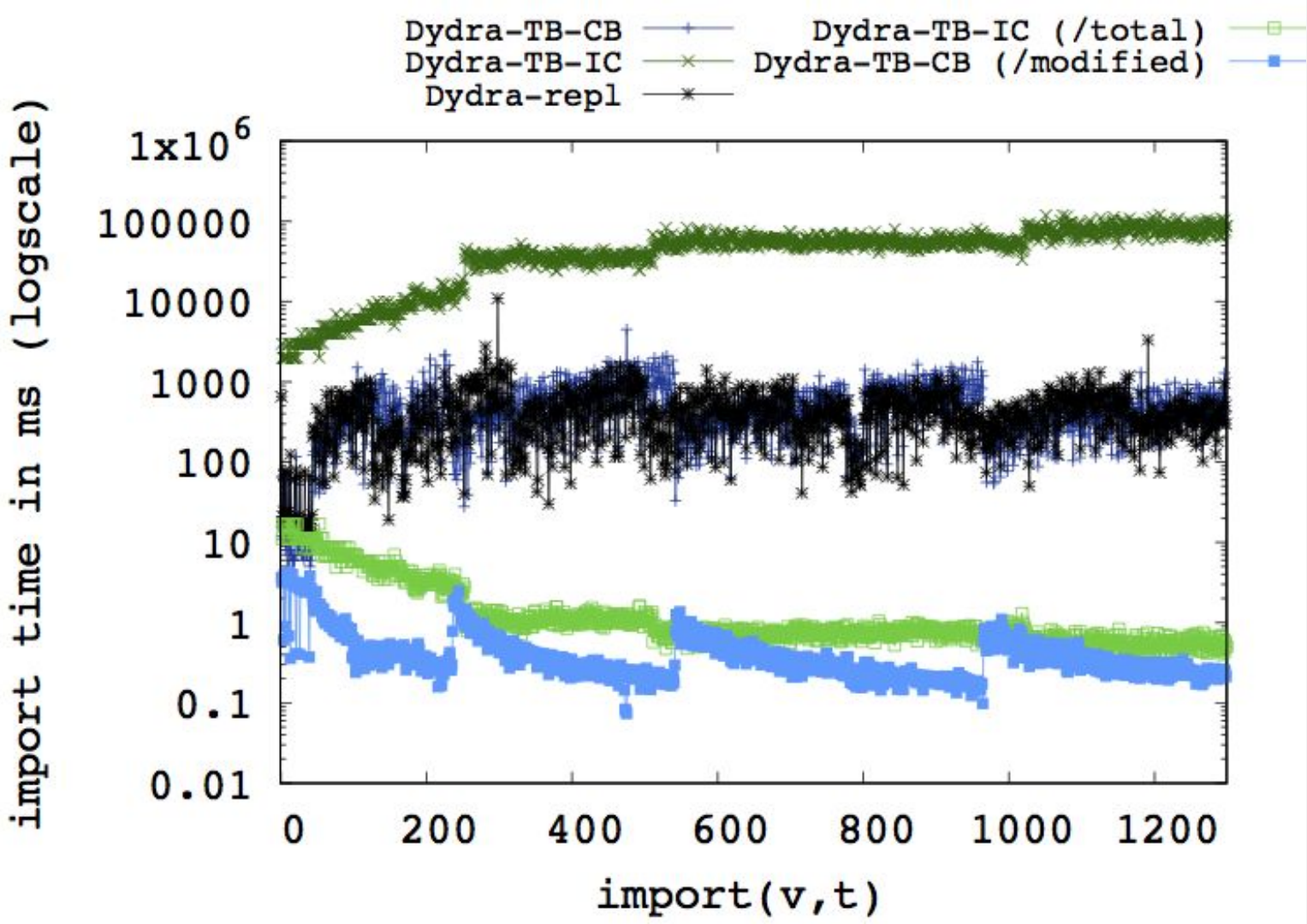
mat, diff, and join retrieval all with performance on the order of "independent copy" storage architectures.

1. intent
2. architecture
3. **import time**
4. import space
5. query time
6. issues ...

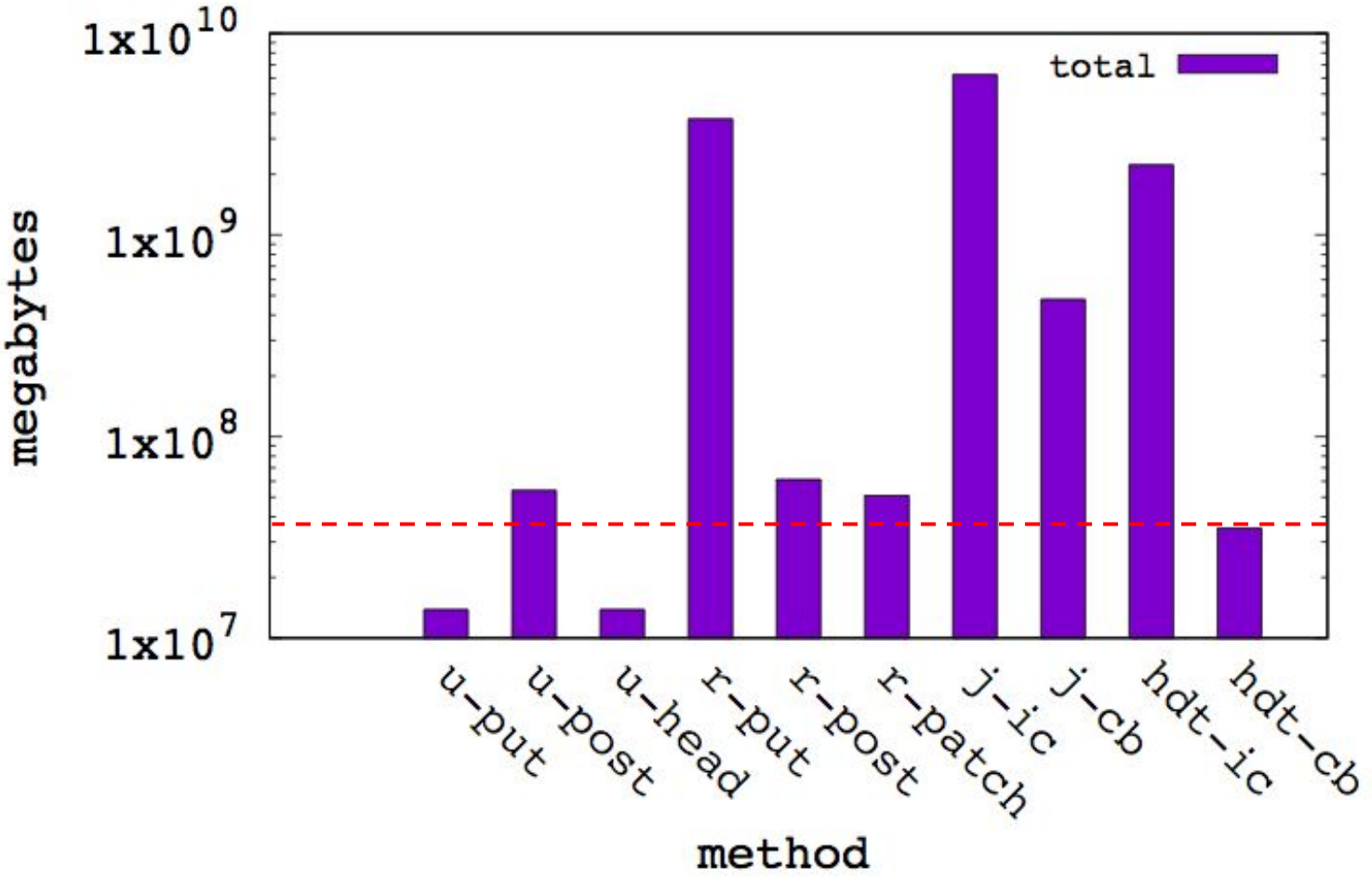


the storage model for replicated data follows the same pattern.
in order to reconcile versions, the visibility map contains universal revision identifiers instead of the local ordinals

1. intent
2. architecture
3. **import time**
4. import space
5. query time
6. issues ...

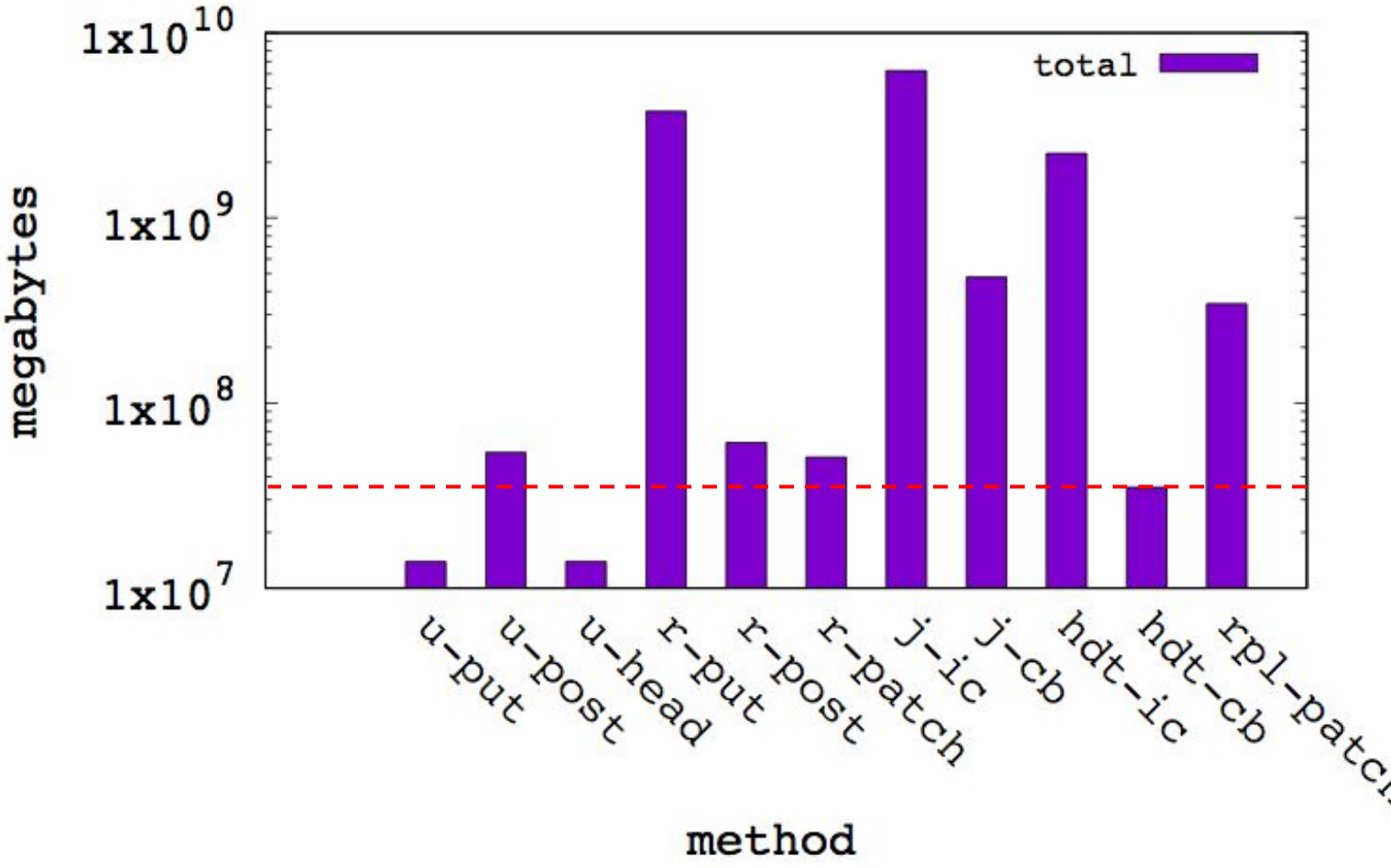


1. intent
2. architecture
3. import time
4. **import space**
5. query time
6. issues ...



1. intent
2. architecture
3. import time
4. **import space**
5. query time
6. issues ...

replication annotations will require garbage collection:



1. intent
2. architecture
3. import time
4. import space
5. query time
6. issues ...

several issues remain open to further investigation

- revision expression algebra
  - implicit v/s explicit : we extract implicit constraints from filters
  - vocabulary : we use a variant for allen intervals
- temporal map representation and compression
  - 77,526,584 v/s 343,512,128 : where the revision identifier relies on references, but retains old instances, its size far exceeds that of an ordinal map, in-place,
- optimization in the context of temporal filters : which cardinalities count
- what access method suits use cases which invert the dependency. for example, temporal social analysis, for which data is stored in temporal slices