#LD

Linked Data Fragments

Querying multiple Linked Data sources on the Web

Ruben Verborgh

UNIVERSITEIT GENT

iMinds

If you have
a Linked Open Data set,
you probably wonder:

"How can people query
my Linked Data on the Web?"

“A public SPARQL endpoint gives live querying, but it’s costly and has availability issues.”

“Offer a data dump. but it’s not really Web querying: users need to set up an endpoint”

“Publish Linked Data documents. But querying is very slow…”

Querying Linked Data on the Web always involves trade-offs.

But have we looked at all possible trade-offs?

Querying Linked Data live on the Web becomes affordable

by building simpler servers and more intelligent clients.

Querying multiple Linked Data sources on the Web

#LD
Linked Data Fragments

Linked Data Fragments

Querying multiple Linked Data sources

Publishing Linked Data at low cost

The Resource Description Framework captures facts as triples.

</articles/www> a schema:ScholarlyArticle.

</articles/www> schema:name "The World-Wide Web".

</articles/www> schema:author </people/timbl>.

</articles/www> schema:author </people/cailliau>.

</articles/www> schema:author </people/groff>.

SPARQL is a language *(and protocol)* to query RDF datasources.

```
SELECT * WHERE {
  ?article a schema:ScholarlyArticle.
  ?article schema:author ?author.
  ?author  schema:name "Tim Berners-Lee".
}
```

Using a data dump, you can set up your own triple store and query it.

Install a local triple store.

Unzip and load all triples into it.

Execute the SPARQL query.

A SPARQL endpoint lets clients execute SPARQL queries over HTTP.

The server has a triple store.

The client sends a query to the server.

The server executes the query and sends back the results.

Querying multiple Linked Data sources on the Web

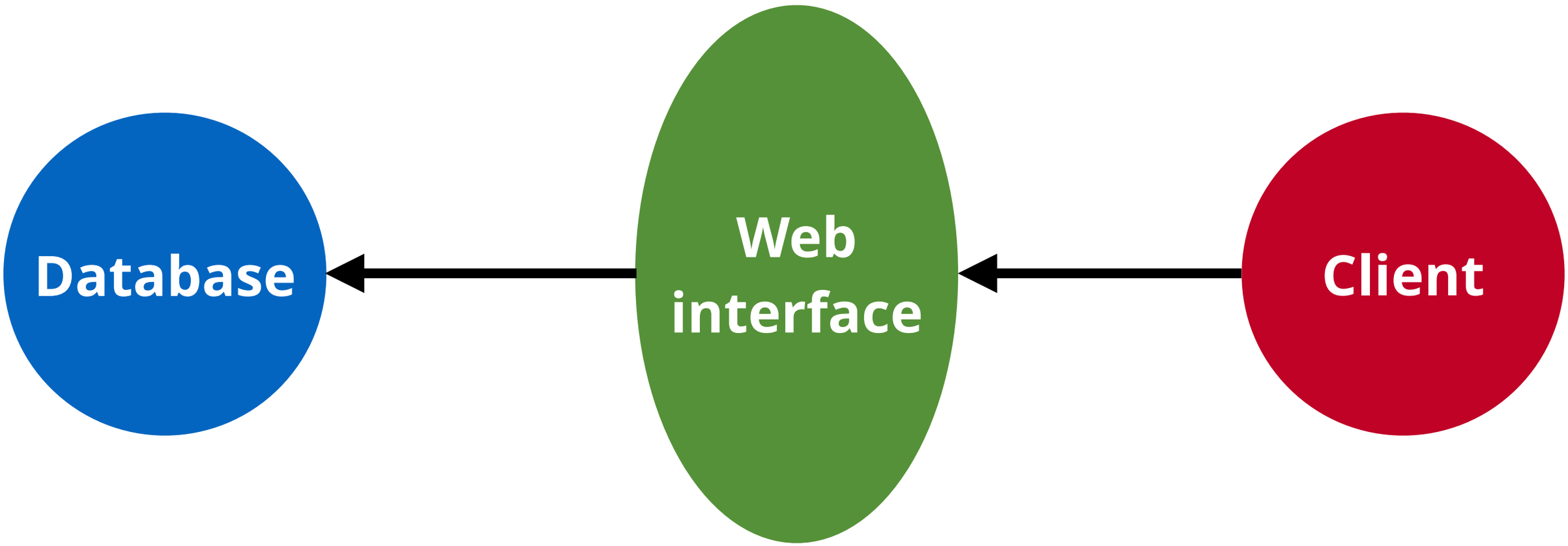
#LD
Linked Data Fragments

Linked Data Fragments

Querying multiple Linked Data sources

Publishing Linked Data at low cost

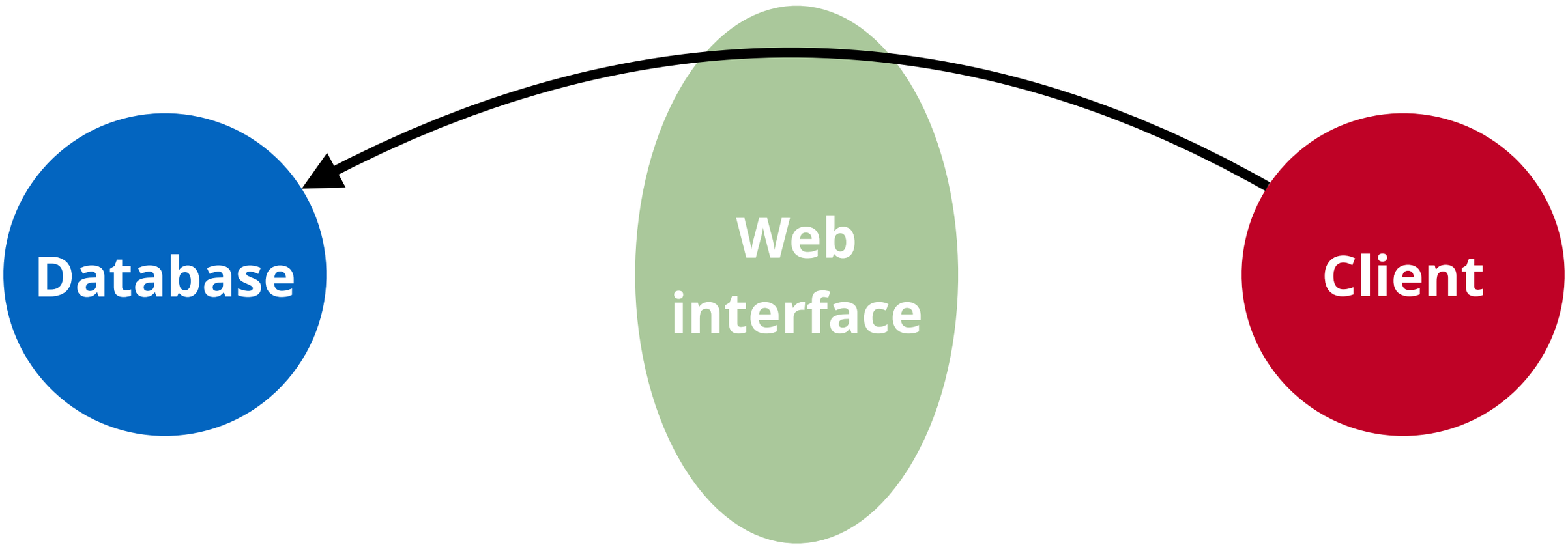
Web interfaces act as gateways between clients and databases.



The interface hides the database schema.

The interface restricts the kind of queries.

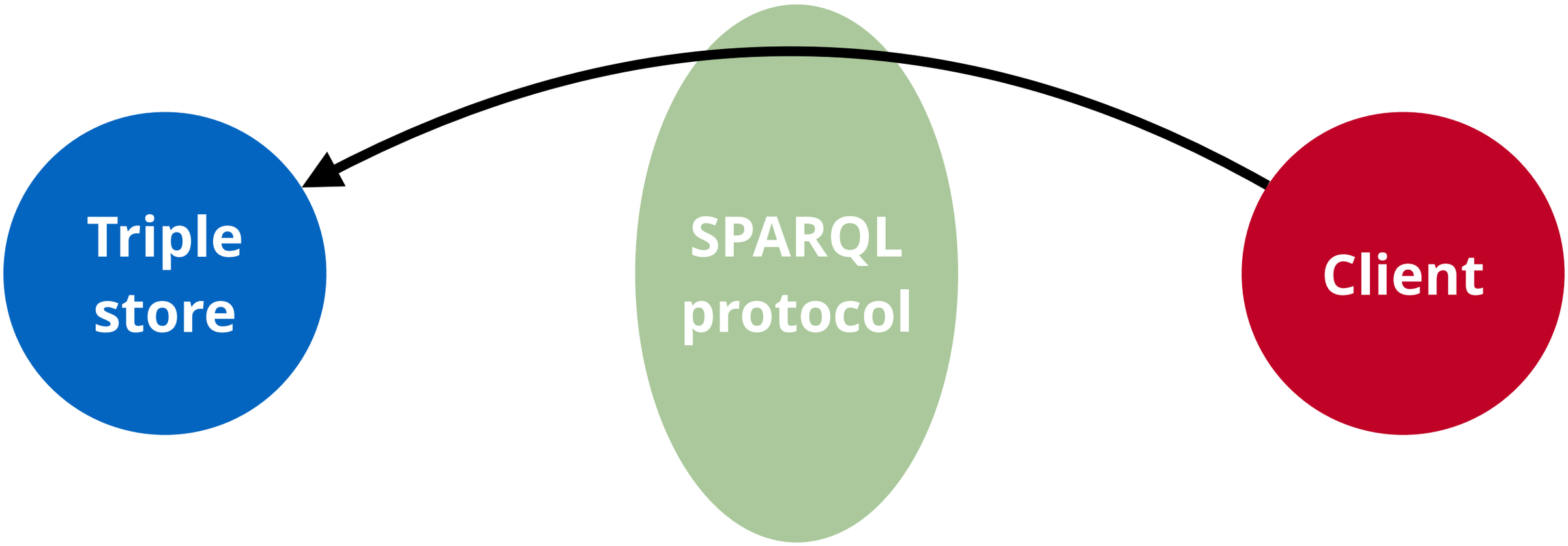
No sane Web developer or admin would give direct database access.



The client must know the database schema.

The client can ask any query.

SPARQL endpoints happily give direct access to the database.



~~The client must know the database schema.~~

The client can ask any query.

Queryable Linked Data on the Web has a two-sided availability problem.

There a few SPARQL endpoints because they are expensive to host.

Those endpoints that are on the Web suffer from frequent downtime.

The average public SPARQL endpoint is down for 1.5 days *each month*.

With multiple SPARQL endpoints, problems become worse.

1 endpoint has 95% availability.

1.5 days down each month

2 endpoints have 90% availability.

3 days down each month

3 endpoints have 85% availability.

4.5 days down each month

Data dumps allow people to set up their own *private* SPARQL endpoint.

Users need a technical background and the necessary infrastructure.

What about casual usage and mobile devices?

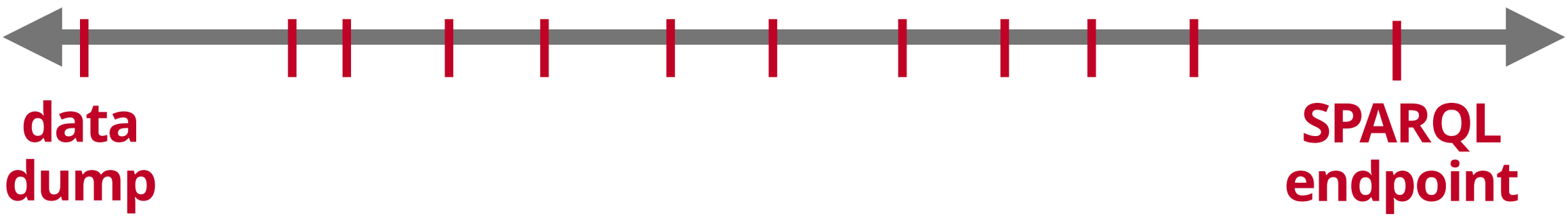
We are not really querying the Web…

It is not an all-or-nothing world.

There is a spectrum of trade-offs.

out-of-date data
high bandwidth
high availability
high client cost
low server cost

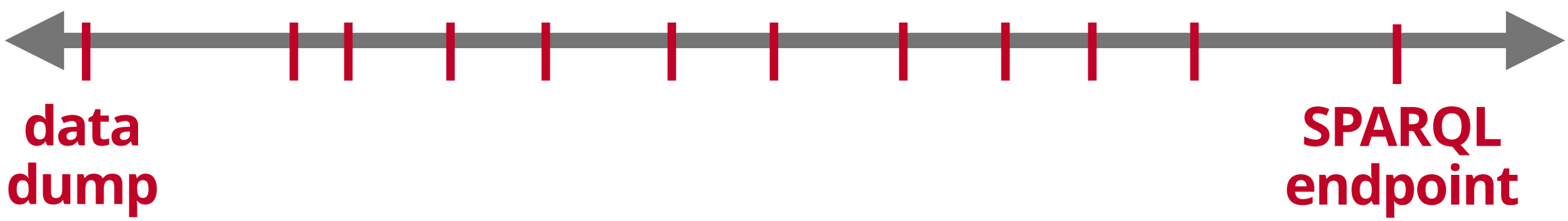
live data
low bandwidth
low availability
low client cost
high server cost



interface offered by the server

Linked Data Fragments **are**
a uniform view on Linked Data interfaces.

Every Linked Data interface
offers specific fragments
of a Linked Data set.



interface offered by the server

Each type of Linked Data Fragment is defined by three characteristics.

data *What triples does it contain?*

metadata *What do we know about it?*

controls *How to access more data?*

Each type of Linked Data Fragment is defined by three characteristics.

data dump

data *all dataset triples*

metadata *number of triples, file size*

controls *(none)*

Each type of Linked Data Fragment is defined by three characteristics.

SPARQL query result

data *triples matching the query*

metadata *(none)*

controls *(none)*

We designed a new trade-off mix with low cost and high availability.

out-of-date data
high bandwidth
high availability
high client cost
low server cost

live data
low bandwidth
low availability
low client cost
high server cost

data dump

SPARQL query results

A Triple Pattern Fragments interface is low-cost and enables clients to query.

live data

high availability

low server cost

A Triple Pattern Fragments interface is low-cost and enables clients to query.

data *matches of a triple pattern (paged)*

metadata *total number of matches*

controls *access to all other fragments*

DBpedia 2014 | DBpedia

fragments.dbpedia.org/2014/en?subject=&predicate=http%3A%2F%2…

DBpedia – Linked Data Fragments

Query DBpedia 2014 by triple pattern

subject:

predicate: dbpedia-owl:birthPlace

object: dbpedia:Italy

Find matching triples

controls *(other fragments)*

Showing triples 1 to 101 of ±8141

metadata *(total count)*

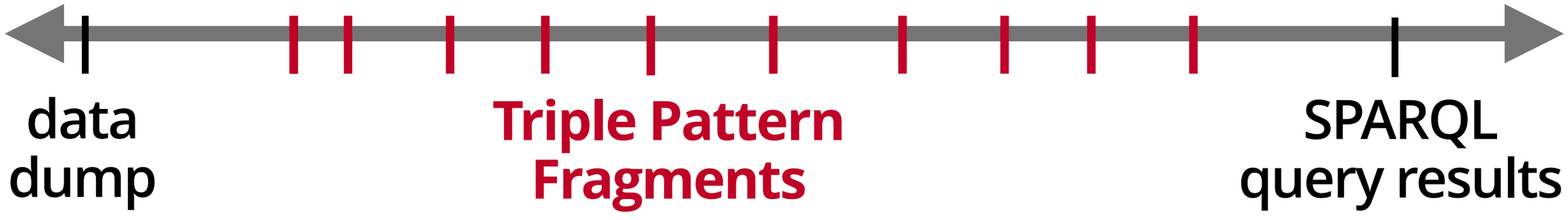
```
%C3%81lvaro_Crespi   birthPlace   Italy.
%C3%89douard_Fachleitner   birthPlace   Italy.
108_(artist)   birthPlace   Italy.
A._F._K._Organski   birthPlace   Italy.
Aaron_March   birthPlace   Italy.
Abdon_Sgarbi   birthPlace   Italy.
Abel_Gigli   birthPlace   Italy.
Abelardo_Olivier   birthPlace   Italy.
Abele_Blanc   birthPlace   Italy.
Achille_Compagnoni   birthPlace   Italy.
```

data *(first 100)*

Triple patterns are not the final answer.

No interface ever will be.

Triple patterns show how far we can get with simple servers and smart clients.



Querying multiple Linked Data sources on the Web

#LD
Linked Data Fragments

Linked Data Fragments

Querying multiple Linked Data sources

Publishing Linked Data at low cost

Experience the trade-offs yourself on the official DBpedia interfaces.

DBpedia data dump

DBpedia Linked Data documents

DBpedia SPARQL endpoint

DBpedia Triple Pattern Fragments

fragments.dbpedia.org

The LOD Laundromat hosts 650.000 Triple Pattern Fragment APIs.

Datasets are crawled from the Web, cleaned, and compressed to HDT.

This shows the potential of a very light-weight interface.

Centralization is not a goal though: we aim for distributed interfaces.

How can intelligent clients
solve SPARQL queries over fragments?

Give them a SPARQL query.
Give them a URL of any dataset fragment.

They look inside the fragment
to see how to access the dataset

and use the metadata
to decide how to plan the query.

Suppose a client needs to evaluate this query against a TPF interface.

```
SELECT  ?person  ?city  WHERE {
    ?person  rdf:type  dbpedia-owl:Scientist.
    ?person  dbpedia-owl:birthPlace  ?city.
    ?city    foaf:name  "Geneva"@en.
}
```

Fragment: http://fragments.dbpedia.org/2014/en

Triple Pattern Fragment servers *enable* clients to be intelligent.

controls *The HTML representation explains: "you can query by triple pattern".*

Triple Pattern Fragment servers *enable* clients to be intelligent.

```
<http://fragments.dbpedia.org/2014/en#dataset> hydra:search [
   hydra:template "http://fragments.dbpedia.org/2014/en
                          {?subject,predicate,object}";
   hydra:mapping
      [ hydra:variable "subject";    hydra:property rdf:subject ],
      [ hydra:variable "predicate"; hydra:property rdf:predicate ],
      [ hydra:variable "object";     hydra:property rdf:object ]
].
```

controls *The RDF representation explains: "you can query by triple pattern".*

Triple Pattern Fragment servers *enable* clients to be intelligent.

Showing triples 1 to 101 of ±8141

```
%C3%81lvaro_Crespi   birthPlace   Italy.
%C3%89douard_Fachleitner   birthPlace   Italy.
108_(artist)   birthPlace   Italy.
```

metadata *The HTML representation explains: "this is the number of matches".*

Triple Pattern Fragment servers *enable* clients to be intelligent.

```
<#fragment>  void:triples  8141.
```

metadata *The RDF representation explains: "this is the number of matches".*

The server has triple-pattern access, so the client splits a query that way.

```
SELECT  ?person  ?city  WHERE {
    ?person  rdf:type  dbpedia-owl:Scientist.
    ?person  dbpedia-owl:birthPlace  ?city.
    ?city    foaf:name  "Geneva"@en.
}
```

Fragment: http://fragments.dbpedia.org/2014/en

The client gets the fragments and inspects their metadata.

?person rdf:type dbpedia-owl:Scientist **18.000**

first 100 triples

?person dbpedia-owl:birthPlace ?city. **625.000**

first 100 triples

?city foaf:name "Geneva"@en. **12**

first 100 triples

Execution continues recursively using metadata and controls.

?person rdf:type dbpedia-owl:Scientist

?person dbpedia-owl:birthPlace ?city.

?city foaf:name "Geneva"@en. **12**

dbpedia:Geneva foaf:name "Geneva"@en.

dbpedia:Geneva,_Alabama foaf:name "Geneva"@en.

dbpedia:Geneva,_Idaho foaf:name "Geneva"@en.

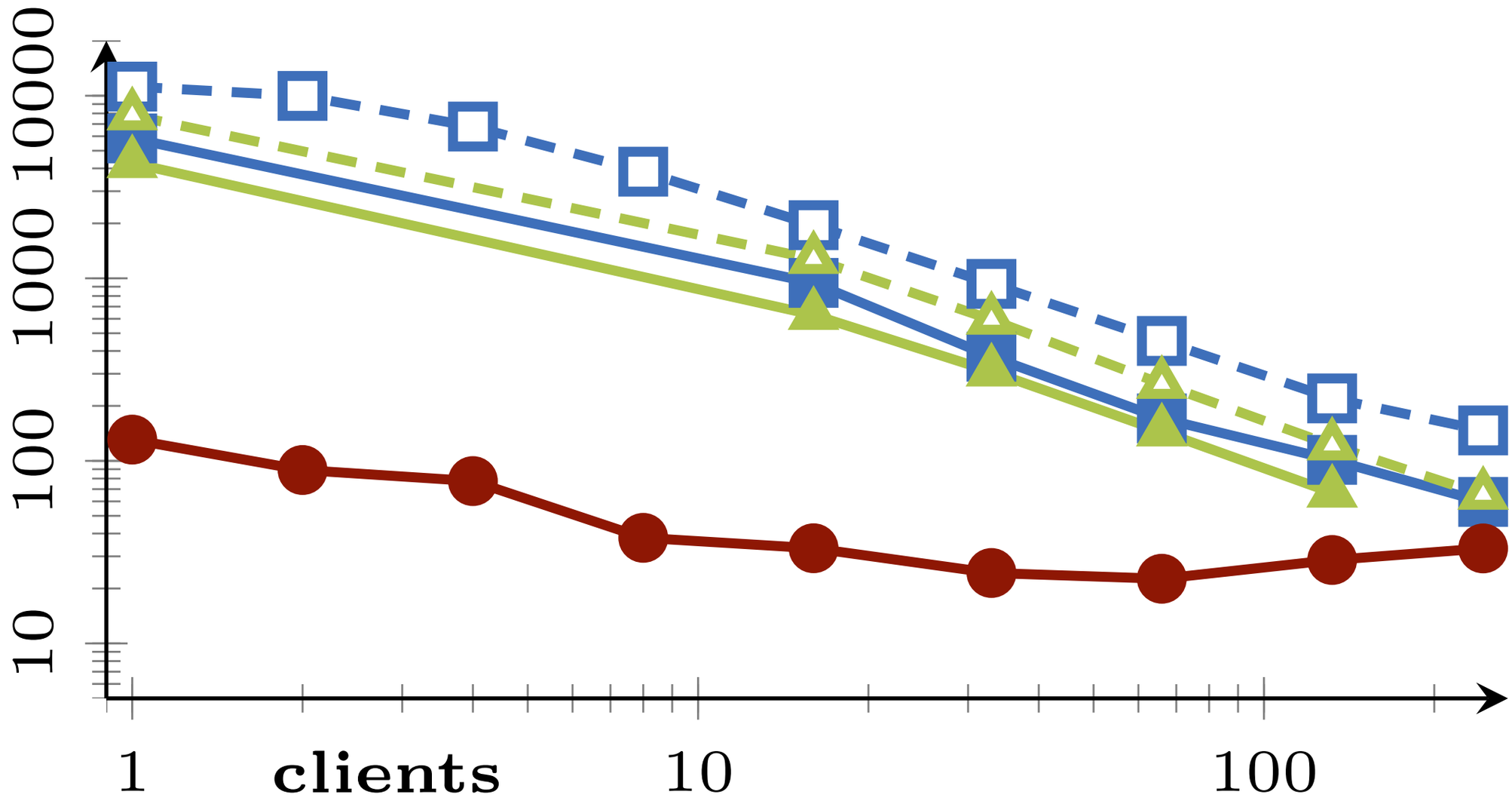
...

Executing this query with TPFs takes 3 seconds—consistently.

```
SELECT  ?person  ?city  WHERE {
    ?person  rdf:type  dbpedia-owl:Scientist.
    ?person  dbpedia-owl:birthPlace  ?city.
    ?city    foaf:name  "Geneva"@en.
}
```

Results arrive in a streaming way, already after 0.5 seconds.

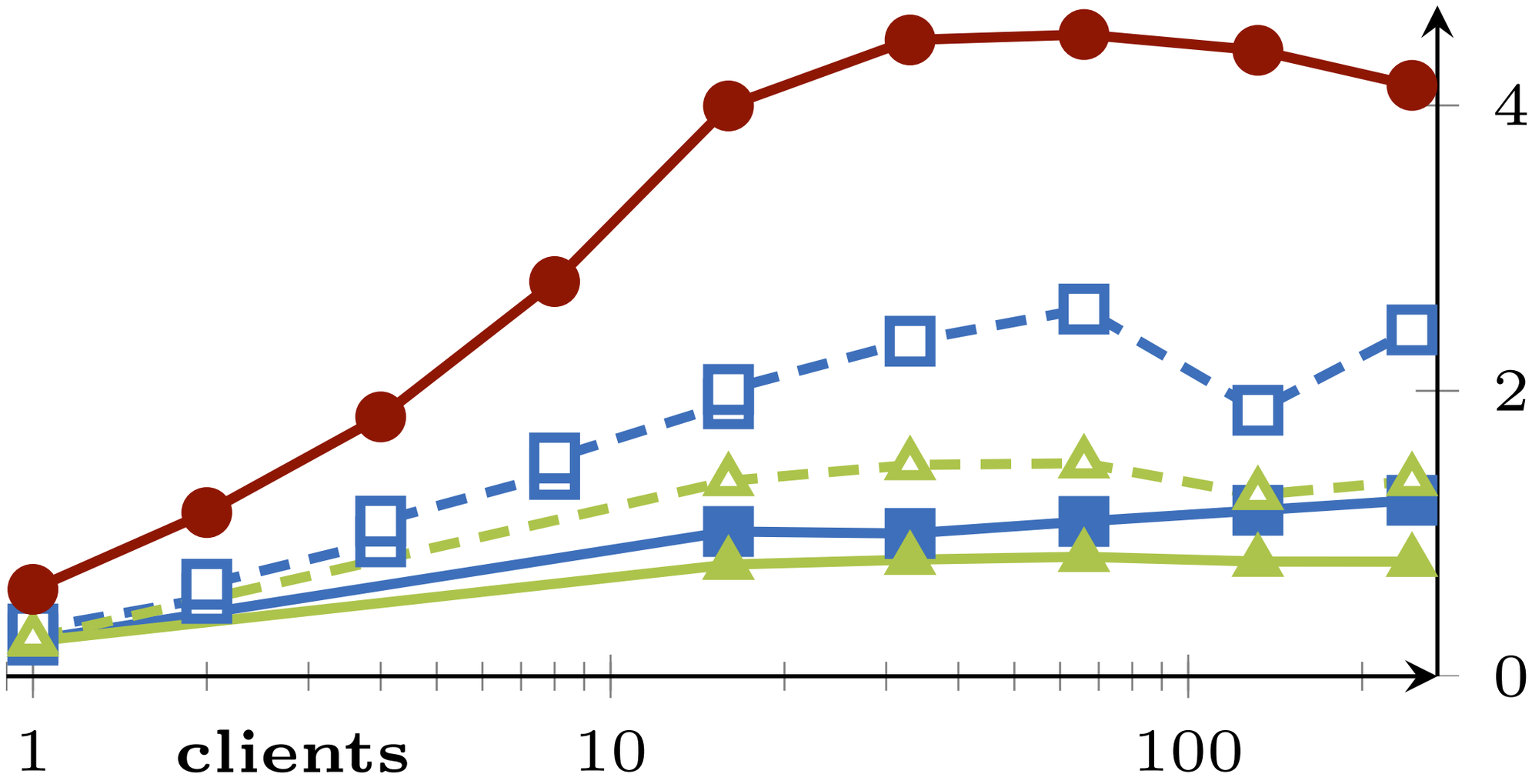
The query throughput is lower,
but resilient to high client numbers.



executed SPARQL queries per hour

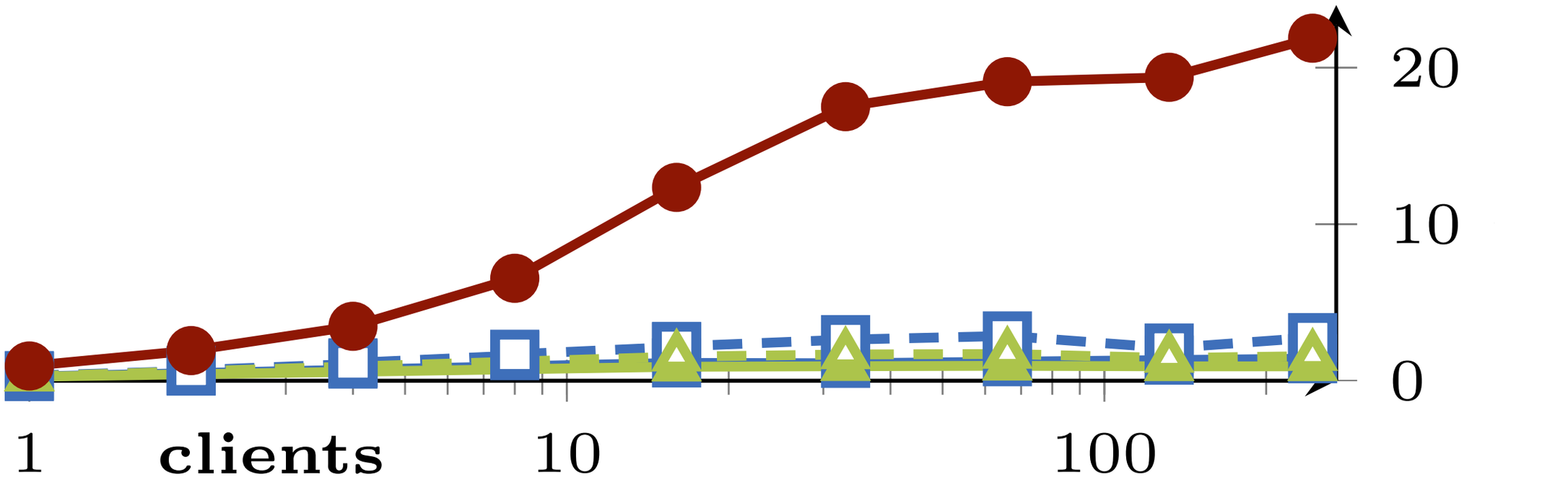
The server traffic is higher,

but requests are significantly lighter.



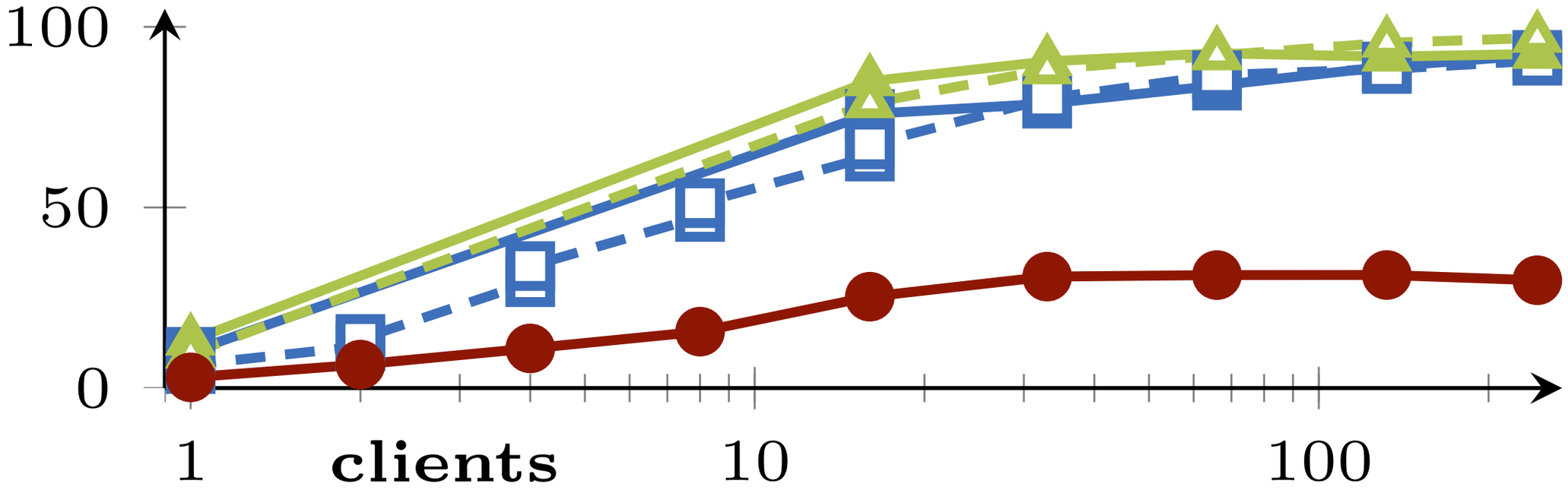
data sent by server in MB

Caching is significantly more effective,
as clients reuse fragments for queries.



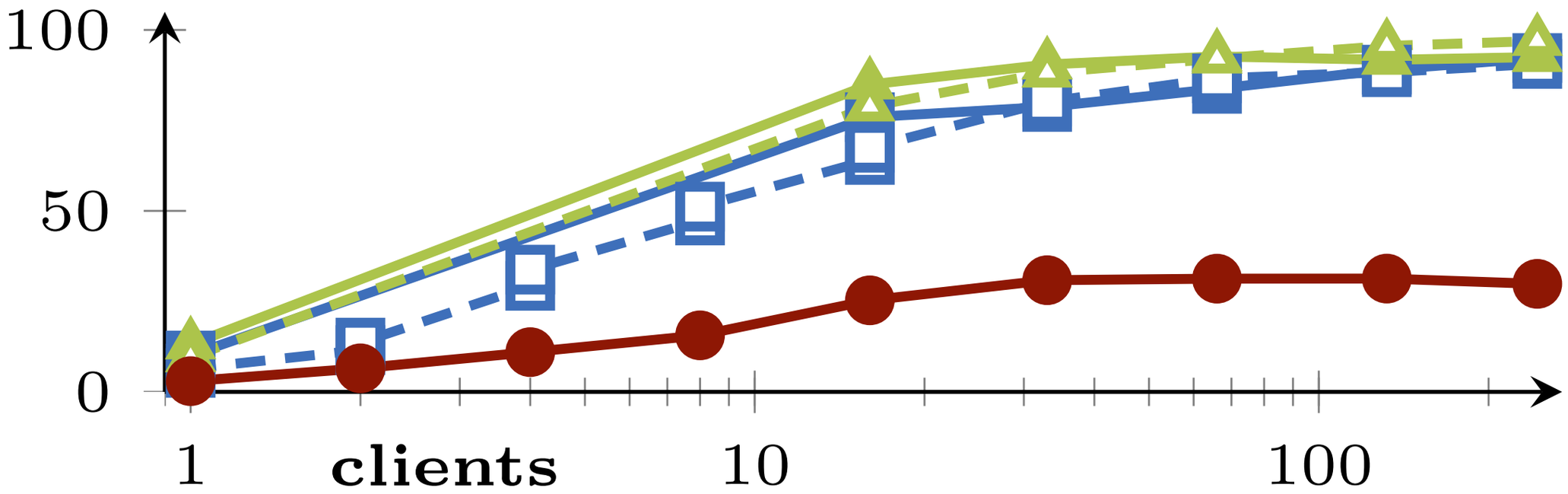
data sent by cache in MB

The server uses much less CPU,
allowing for higher availability.



server CPU usage per core

Servers enable *clients* to be intelligent,
so they remain simple and light-weight.



server CPU usage per core

Querying multiple Linked Data sources on the Web

#LD
Linked Data Fragments

Linked Data Fragments

Querying multiple Linked Data sources

Publishing Linked Data at low cost

Triple Pattern Fragments publication is absolutely straightforward.

Servers only need to implement a simple API.

A SPARQL endpoint as backend is not a necessity.

The compressed HDT format is very fast for triple patterns.

All software is available

as open source.

Software

github.com/LinkedDataFragments

Documentation and specification

linkeddatafragments.org

Publishing a Linked Dataset involves only three steps.

Convert your dataset to the compressed HDT format.

Configure your dataset in the LDF server.

Expose the LDF server on the public Web.

Convert your dataset to HDT

for fast triple pattern lookups.

rdf2hdt -f turtle -i dataset.ttl -o dataset.hdt

or *http://lodlaundromat.org/basket/*

Install an LDF server

and configure your datasource.

```
# install through Node.js
npm install -g ldf-server

# run 4 workers on port 5000
ldf-server config.json 5000 4
```

Install an LDF server
and configure your datasource.

```
{
  "title": "My Linked Data Fragments server",
  "datasources": {
    "dbpedia": {
      "title": "DBpedia 2015",
      "type": "HdtDatasource",
      "description": "DBpedia 2015 with an HDT back-end",
      "settings": { "file": "data/dbpedia2015.hdt" }
    }
  }
}
```

Set up a public Web server ("reverse proxy") with caching.

You can run the LDF server directly on port 80.

Alternatively, use Apache or NGINX as a proxy/cache in front.

Set up a public Web server ("reverse proxy") with caching.

```
server {
  server_name data.example.org;

  location / {
    proxy_pass http://127.0.0.1:5000$request_uri;
    proxy_set_header Host $http_host;
    proxy_pass_header Server;
  }
}
```

…or again, just

http://lodlaundromat.org/basket/

;-)

Querying multiple Linked Data sources on the Web

#LD
Linked Data Fragments

Linked Data Fragments

Querying multiple Linked Data sources

Publishing Linked Data at low cost

#LD

Linked Data Fragments

@LDFragments

@RubenVerborgh