Triple Stores
What is a triple store?

- A database for RDF triples
- Can ingest RDF in a variety of formats
- Supports a query language
  - SPARQL is the W3C recommendation
  - Other RDF query languages exist (e.g., RDQL)
  - Might or might not do inferencing
- Triples stored in memory in a persistent backend
- Persistence provided by a relational DBMS (e.g., mySQL) or a custom DB for efficiency.
Architectures

- Can be divided into several categories: *In-memory*, *Native store*, *Non-native store*
- In memory: RDF Graph is stored as triples in main memory
- Native store: Persistent storage systems with custom DBs, e.g.: JENA TDB, Sesame Native, Virtuoso, AllegroGraph, Oracle 11g
- Non-Native store: Persistent storage systems set-up to run on third party DBs, e.g., Jena SDB using mysql or postgres
Architecture trade-offs

- In memory is fastest, obviously, but load time has to be factored in.
- Native stores are fast, scalable, and popular now.
- Non-native stores may be better if you have a lot of updates and/or need good concurrency control.
- See the W3C page on large triple stores for some data on scaling for many stores.
Large triple stores in 2018

1. Oracle Spatial and Graph with Oracle Database 12c - 1.08 Trillion triples (edges)
2. AllegroGraph (1+Trillion)
3. Stardog (50B)
4. OpenLink Virtuoso v7+ - 39.8B+ explicit; uncounted virtual/inferred
   4.1 Benchmarks data sources
   4.2 Older comments
5. GraphDB™ by Ontotext (17B)
   5.1 Performance Benchmark Results
   5.2 Detailed Benchmark Study
   5.3 Notes
6. Garlik 4store (15B)
7. Bigdata(R) (12.7B)
8. YARS2 (7B)
9. Jena TDB (1.7B)
10. RDFox (1.6B)
11. Others who claim to go big
12. Questions
   12.1 Jena SDB (650M)
   12.2 Mulgara (500M)
   12.3 RDF gateway (262M)
   12.4 Jena with PostgreSQL (200M)
   12.5 Kowari (160M)
   12.6 3store with MySQL 3 (100M)
   12.7 Sesame (70M)

http://www.w3.org/wiki/LargeTripleStores
Many triple stores support quads for **named graphs**

A named graph is just an RDF with a URI name often called the **context**

Such a triple store divides its data a default graph and zero or more additional named graphs

SPARQL has support for named graphs

De facto standards exist for representing quad data, e.g., **n-quads** and **TriG** (a turtle/N3 variant)

**AllegroGraph** stores quints (S,P,O,C,ID), the ID can be used to attach metadata to a triple
Support for Reasoning

- Most triple stores don’t do much (or any) reasoning and use a simple model:
  - You do the reasoning to materialize all of the triples you want, which you then load into the store
  - Triple store provides query and update APIs, access control, SPARQL interface, efficient indexing, etc.

- Some do support reasoning, e.g.,
  - Jena has a native rules engine and an API for external reasoners (e.g., Pellet, Fact++)
  - Sesame has a native RDFS reasoner
  - Stardog supports OWL DL reasoning via query expansion
Example: Jena Framework

- An open software Java system originally developed by HP (2002-2009)
  - Moved to Apache when HP Labs discontinued its Semantic Web research program
  - [https://jena.apache.org/](https://jena.apache.org/)
- Using the TDB native store, it can easily handle ~2B triples
- Good tutorials and documentation
- Has internal reasoners and can work with DIG compliant reasoners such as Pellet
- Supports a Native API and SPARQL via Fuseki
Example: Sesame

- Sesame is an open source RDF framework with support for RDFS inferencing and querying
- http://www.openrdf.org/
- Implemented in Java
- Query languages: SeRQL, RQL, RDQL and SPARQL
- Triples can be stored in memory, on disk, or in a RDBMS
- Has a native RDFS reasoner
- Easy to setup & use, but tops out at ~70M triples
Example: Stardog

- [http://stardog.com/](http://stardog.com/) by Clark and Parsia
- Pure Java RDF database ("quad store")
- Lightweight and very fast for in-memory use
- Reasoning support via Pellet for OWL DL and query rewriting for OWL 2 QL, EL & RL
- Command line interface and JAVA API
- Commercial, but has a free version good for modest projects
- ~50B triples on $10K server with 256G ram and 32 cores
Performance

- Much work on benchmarking of triple stores
- There are several standard benchmark sets
- Two key things are measured include
  - Time to load and index triples
  - Time to answer various kinds of SPARQL queries
- The **Berlin SPARQL Benchmarks** evaluated 4store, BigData, BigOwlim, Jena TDB and Virtuoso in 2011 with 100M and 200M datasets.
- The numbers are “query mixes per hour”, so bigger is better
## Load Time

<table>
<thead>
<tr>
<th>SUT</th>
<th>100M</th>
<th>200M</th>
</tr>
</thead>
<tbody>
<tr>
<td>4store</td>
<td>26:42*</td>
<td>1:12:04*</td>
</tr>
<tr>
<td>BigData</td>
<td>1:03:47</td>
<td>3:24:25</td>
</tr>
<tr>
<td>BigOwlim</td>
<td>17:22</td>
<td>38:36</td>
</tr>
<tr>
<td>TDB</td>
<td>1:14:48</td>
<td>2:45:13</td>
</tr>
<tr>
<td>Virtuoso</td>
<td>1:49:26**</td>
<td>3:59:38**</td>
</tr>
</tbody>
</table>

* The N-Triples version of the dataset was used.

** The dataset was split into 100 respectively 200 Turtle files and loaded with the DB.DBA.TTLP function consecutively.
6.1.1 QMpH: Explore use case

The complete query mix is given here.

<table>
<thead>
<tr>
<th></th>
<th>100m</th>
<th>200m</th>
</tr>
</thead>
<tbody>
<tr>
<td>4store</td>
<td>5589</td>
<td>4593</td>
</tr>
<tr>
<td>BigData</td>
<td>2428</td>
<td>1795</td>
</tr>
<tr>
<td>BigOwlim</td>
<td>3534</td>
<td>1795</td>
</tr>
<tr>
<td>TDB</td>
<td>2274</td>
<td>1443</td>
</tr>
<tr>
<td>Virtuoso</td>
<td>7352</td>
<td>4669</td>
</tr>
</tbody>
</table>

A much more detailed view of the results for the Explore use case is given under [Detailed Results For The Explore use case](#).

6.1.2 QMpH: Explore and Update use case

The Explore and Update query mix consists of the [Update query mix](#) (queries 1 and 2) and the [Explore query mix](#) (queries 3 and 4).

<table>
<thead>
<tr>
<th></th>
<th>100m</th>
</tr>
</thead>
<tbody>
<tr>
<td>4store</td>
<td>5311</td>
</tr>
<tr>
<td>BigOwlim</td>
<td>2809</td>
</tr>
<tr>
<td>TDB</td>
<td>680</td>
</tr>
</tbody>
</table>
A triple store is an essential component of any system using RDF.

There are a number of good ones available, both open sourced and commercial.

Developing triple stores for large-scale parallel systems is still a research topic.