Triple Stores
What is a triple store?

- A specialized 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
  - Most query languages don’t handle inserts
- Triple stored in memory in a persistent backend
- Persistence provided by a relational DBMS (e.g., mySQL) or a custom DB for efficiency.
Architectures

- Based on their implementation, can be divided into several broad 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 their own implementation of databases. E.g., JENA TDB, Sesame Native, Virtuoso, AllegroGraph, Oracle 11g

- **Non-Native store**: Persistent storage systems set-up to run on third party DBs. Eg. 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

1 AllegroGraph (1 +Trillion)

2 OpenLink Virtuoso v6.1 - 15.4B+ explicit; uncounted virtual/inferred
   2.1 Benchmarks data sources
   2.2 Older comments

3 BigOWLIM (12B explicit, 20B total): 100,000 queries per $1
   3.1 Scalability and Loading Speed
   3.2 Query Performance, Horizontal Scalability in the Cloud
   3.3 Performance features

4 Garlik 4store (15B)

5 Bigdata(R) (12.7B)

6 YARS2 (7B)

7 Jena TDB (1.7B)

8 Jena SDB (650M)

9 Mulgara (500M)

10 RDF gateway (262M)

11 Jena with PostgreSQL (200M)

12 Kowari (160M)

13 3store with MySQL 3 (100M)

14 Sesame (70M)

15 Others who claim to go big

16 Questions

17 Related
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 quint (S,P,O,C,ID), the ID can be used to attach metadata to a triple.
Example: Jena Framework

- An open software Java system originally developed by HP (2002-2009)
  - [http://incubator.apache.org/jena/](http://incubator.apache.org/jena/)
- Moved to Apache when HP Labs discontinued its Semantic Web research program ~2009
- Good tutorials
- Has internal reasoners and can work with DIG compliant reasoners or Pellet.
- Supports a Native API and SPARQL
- Joseki is an add-on that provides a SPARQL endpoint via an HTTP interface.
Jena Features

- API for reading, processing and writing RDF data in XML, N-triples and Turtle formats;
- Ontology API for handling OWL and RDFS ontologies;
- Rule-based inference engine for reasoning with RDF and OWL data sources;
- Stores to allow large numbers of RDF triples to be efficiently stored on disk;
- Query engine compliant with the latest SPARQL specification
- Servers to allow RDF data to be published to other applications using a variety of protocols, including SPARQL
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

Triples can be stored in memory, on disk, or in a RDBMS
Example: Stardog

- [http://stardog.com/](http://stardog.com/) by Clark and Parsia
- Pure Java RDF database (“quad store”)
- Designed to be lightweight and very fast for in memory stores
- Performance for complex SPARQL queries
- Reasoning support via Pellet for OWL DL and query rewriting for OWL 2 QL, EL & RL
- Command line interface and JAVA API
Can we build efficient triple stores around conventional RDBMS technology?

What are the performance issues?
- Load time?
- Interfencing?

How well does is scale?
A lot of work has been done on benchmarking triples 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

See, for example, recent (2011) data from the Berlin SPARQL Benchmarks which studied 4store, BigData, BigOwlim, TDB and Virtuoso.
## 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 fiiles 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 through 5).

<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>