



Balancing Between Scalable Repository and Light-Weight Reasoner

WWW2006, Edinburgh

25 May, 2006

BG Facts



Bulgaria is a small country on the Balkans, i.e. in Europe

Still, we get surprisingly good results in unexpected areas:

- **Chess**: Bulgarians hold all the major titles:
 - **Vesselin Topalov**: World Champion in Chess since Oct 2005
 - **Antoaneta Stefanova**: Woman's World Champion since 2004
- **Ice Dancing Pairs**: Denkova and Staviyski, World Champions since March'06
- **Sumo Fight**: Kotooshu (Kaloyan Mahlyanov) is making a record-breaking career in Japan
- Bulgaria hosted the **first Internat. Olympiad in Informatics**, 1989
- John Atanasoff is the inventor for **the first "electronic" computer** (1937-1939), together with Clifford Berry in ... Iowa

It is not so strange, that Bulgarians developed
The fastest OWL repository in the world!

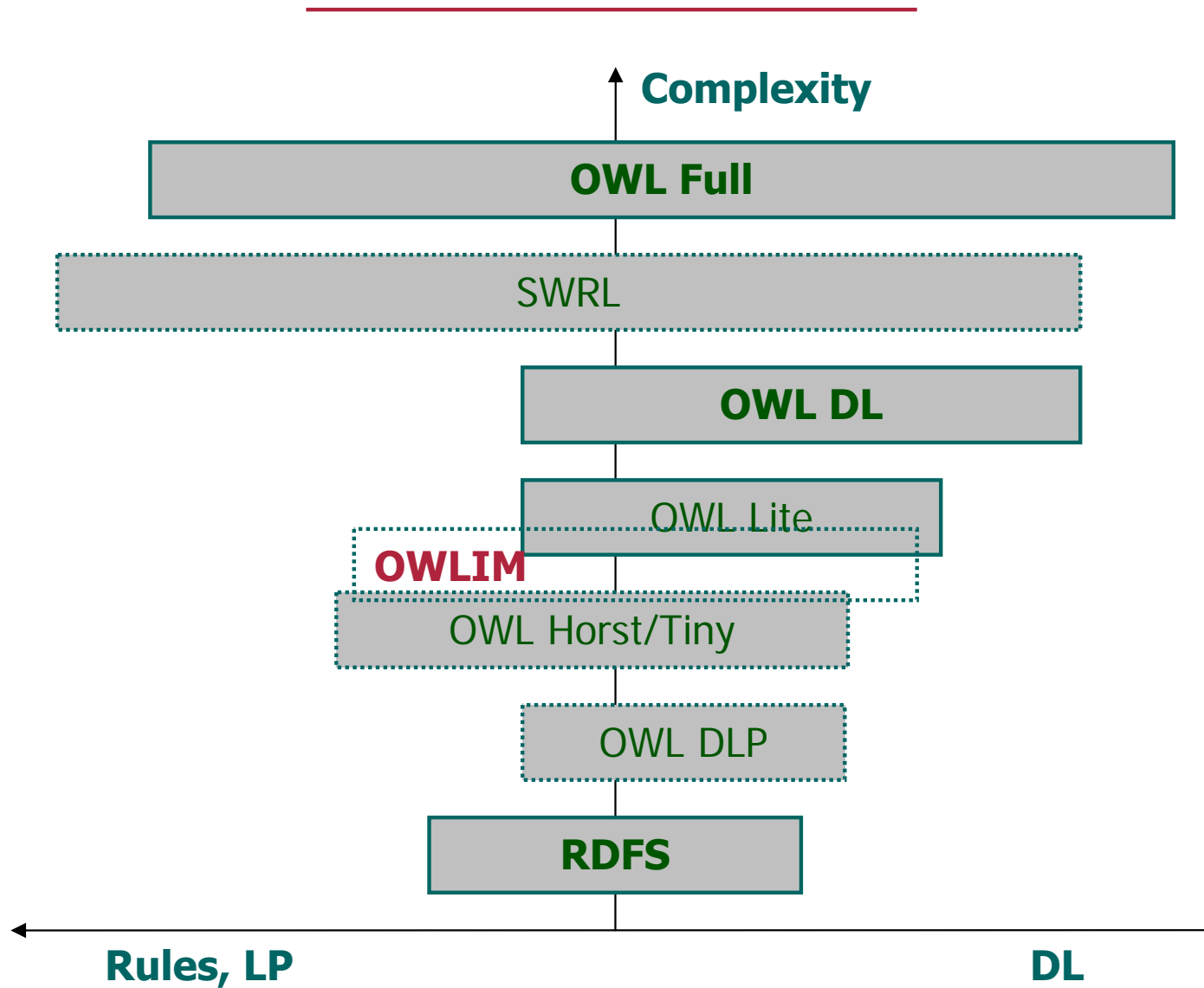
Semantic Repository Supporting OWL

- OWLIM is a **scalable semantic repository** which allows
 - Management, integration, and analysis of heterogeneous data
 - combined with light-weight reasoning capabilities
- Its performance and scalability allow it to **replace RDBMS** in a wide range of applications
 - It is suitable for analytical tasks and Business Intelligence (OLAP)
 - Not suitable for highly dynamic transaction-oriented environments (OLTP)
- OWLIM provides full support for **RDF(S)** and limited **OWL Lite**
 - It supports OWL Horst/Tiny, which is more expressive than OWL DLP
 - **Rule-extensions** are also possible, as it is based on a rule engine

OWL Horst

- Herman ter Horst, H. J. *Combining RDF and Part of OWL with Rules: Semantics, Decidability, Complexity*. In Proc. of ISWC 2005.
- Defines **R-Entailment**: entailment over RDF graph based on rules of triple patterns
 - Rules are defined through generalized-triple patterns
 - Which may contain variable at any position
 - ter Horst proves R-Entailment has “better” complexity and decidability than Datalog-based OWL mappings and rule extensions.
- Defines **pD* entailment** rules, which:
 - Go beyond OWL DLP, dropping the DL-specific constraints
 - Are fully compatible with RDFS (including meta-modelling)
 - Extend the RDF(S) semantics to assure better handling of the typed literals

Naïve OWL Fragments Map



Semantics Supported by OWLIM

- The reasoning support in OWLIM is customizable;
- The **ruleset** parameter allows for switching between 4 predefined inference modes:
 - **owl-max** – the most expressive set (see the next slides);
 - **owl-horst** – a set similar to the one defined in [Horst05]:
 - It is sufficient to pass the LUBM benchmark correctly;
 - Similar to what was defined as OWL-Tiny at SWAD-Europe'03
 - **rdfs** – the standard RDF(S) semantics;
 - **empty** – as an RDF store without any inference;
- The **partialRDFS** parameter allows switching on/off an optimization in the RDFS support (see slide: RDFS Support)

OWL Support

- OWLIM supports almost all the OWL primitives (in owl-max mode):
 - No specific support for: Thing, Nothing, differentFrom, complementOf
- The semantics supported is **close to OWL Lite**
 - Ignoring (min/max)Cardinality with values greater than 1
- Major differences from OWL Lite:
 - The semantics of some primitives is only partially supported
 - See the System Doc. and the **rules.pie** file, which contains all the axioms and rules used (the reasoning options are clearly indicated)
 - There are **no DL-specific constraints**; the following cases are OK:
 - Meta-classes (custom classes of classes);
 - Properties linking classes and instances are OK
- All the relevant normative entailment tests from “OWL Tests” pass correctly (available as JUnit tests in the distribution)

RDFS Support

- The standard RDF(S) semantics, [Hayes 04], is supported
 - Except some D-entailment related to typed literals, which are omitted for performance reasons
- Optimized “partial” RDFS option is available for better performance. In this mode:
 - some “trivial” RDFS axioms are excluded;
 - `<X, rdf:type, rdf:Resource>` statements are not being inferred for all subject and predicates;
 - Rationale: most of the applications do not need such inference. With `partialRDFS=true`:
 - LUBM benchmark queries are evaluated properly;
 - KIM (using the PROTON ontology) works properly.
 - This optimization is irrelevant when `ruleset=empty`.

Language Support Comparison

- The **owl-max** semantics of OWLIM is **close to OWL Horst**:
 - owl-max is generally richer than the pD^* -entailment of Horst, but
 - OWLIM has no support for the inconsistency rules in pD^*
 - No datatype-supporting modifications to the RDFS semantics (D^*)
- OWLIM (owl-max) vs. OWL DLP
 - OWLIM supports a fragment **richer than OWL DLP**
 - OWLIM supports the full RDFS semantics, while DLP considers the DL-specific constraints
- OWLIM vs. OWL Lite:
 - OWLIM supports the full RDFS semantics, while OWL Lite does not
 - Incomplete support for some primitives

In-memory Reasoning and Reliable Persistence

- OWLIM uses TRREE (Triple Reasoning and Rule Entailment Engine)
 - TRREE implements **R-Entilement**
 - for **forward-chaining** and “total materialization”
- It performs **in-memory reasoning and query evaluation**
- Combined with **reliable persistence** strategy based on N-Triples
- The compromise: **relatively slow delete** operation
 - “limited” scalability on scenarios with high implicit/explicit statements ratio. This does not seem to be the case with many of the popular ontologies in RDFS and OWL
- **Very fast upload, retrieval, query evaluation** for huge ontologies and KB
 - Database-like query optimizations are much easier

A Configurable SAIL for Sesame

- OWLIM is available as a Storage and Inference Layer (SAIL) for **Sesame RDF database** (v.1.2.1-1.2.4). Benefits:
 - Sesame's infrastructure, documentation, user community, etc.
 - Support for multiple query languages (RQL, RDQL, SeRQL)
 - Support for import and export formats (RDF/XML, N-Triples, N3)
- **OWLIM Configuration Options:**
 - **noPersist**: the N-Triples persistency switched off
 - **Configurable semantics**: through the **ruleset** and **partialRDFS** parameters (already presented)
 - **Configurable index-size**: allows trading memory for performance
 - **stackSafe**: switches on a slower mode of the TRREE engine, which practically eliminates the possibility of stack overflow errors

OWLIM Default Configuration

- **OWLIM's default configuration is:**

- **noPersist=false** (i.e. it does store the content of the repository)
- Rule-set/semantics: **owl-horst**
- **partialRDFS=true**
- Index-Size: **4M entries** (64MB of RAM; 16 bytes per entry)
- **stackSafe=false**

- The configuration of OWLIM which provides the same functionality as the Sesame's most popular in-memory **RDFS** **SchemaSail** is:

- **noPersist=true;**
- Rule-set/semantics: **RDFS**
- **partialRDFS=false**

Performance Evaluation Configurations

Name	Configuration	RAM (-Xmx)	JDK	Comment
4cOpt12g	2 x Opteron 270 (2.0GHz, dual-core), Suse Linux v.10, 64-bit	12GB, DDR400	JDK 1.5 64-bit	A database/application server; SATA2 drives; RAID10; ~4000 EURO
2Opt6.0g	2 x Opteron 246 (2.0GHz) Windows Server 2003 64-bit	6GB, DDR400	JDK 1.5 64-bit	A database/application server; SATA2 drives; RAID10; ~3000 EURO
Pdc1.6g	Pentium D 920 (2.8GHz, dual-core), Win XP	1.6GB, DDR2 667	JDK 1.5	Workstation, RAID1 SATA2 drives
Piv0.9g	Pentium IV 630 (3.0GHz), Win XP	900MB, DDR2 533	JDK 1.5	Office desktop
Pm0.7g	Pentium Mobile 1.6GHz, Win XP	680MB, DDR266	JDK 1.5	Notebook (Q2'03)

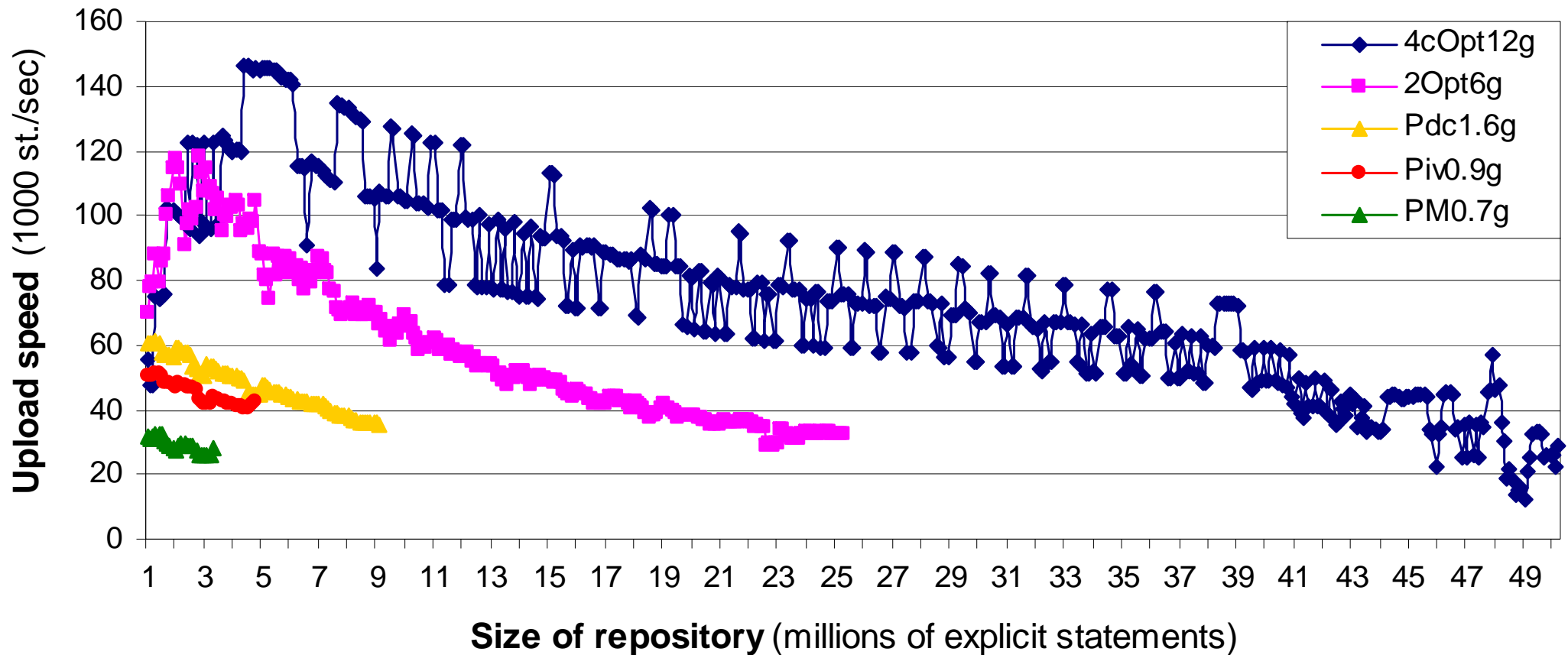
City Benchmark

- The repository is **pre-populated with about 0.5M explicit statements**:
 - a real ontology (PROTON) and
 - a knowledge base – the small version of KIM's World Knowledge Base;
- **Synthetic descriptions of cities** are added incrementally,
 - each transaction adds one city described in about **10k statements**.
- **Interlinking to the non-synthetic part**:
 - the cities are linked to real provinces (randomly chosen from the WKB)
 - 10 synthetic organizations are created and “located” into each city;
 - 38 persons are created and settled within each of the organizations;
 - This way WKB is extended with LDAP-like data in realistic manner.
- A couple of test queries (in SeRQL) are evaluated after the addition of each 10 cities (i.e. after each new 100k statements).

Delete Operation (City Benchmark)

- Delete is also tested on each 100 cities generated
- The time for finalization of a delete transaction on Piv0.9g machine varies with respect to the size of the repository as follows:
 - 19 sec. for 1.5M st.;
 - 31 sec. for 2.5M st.;
 - 43 sec. for 3.5M st.
- As it can be expected due to the straightforward invalidation of the inferred closure:
 - The delete operation is relatively slow;
 - **delete time grows linearly** with about 10 sec. for each new million of statements in the repository.

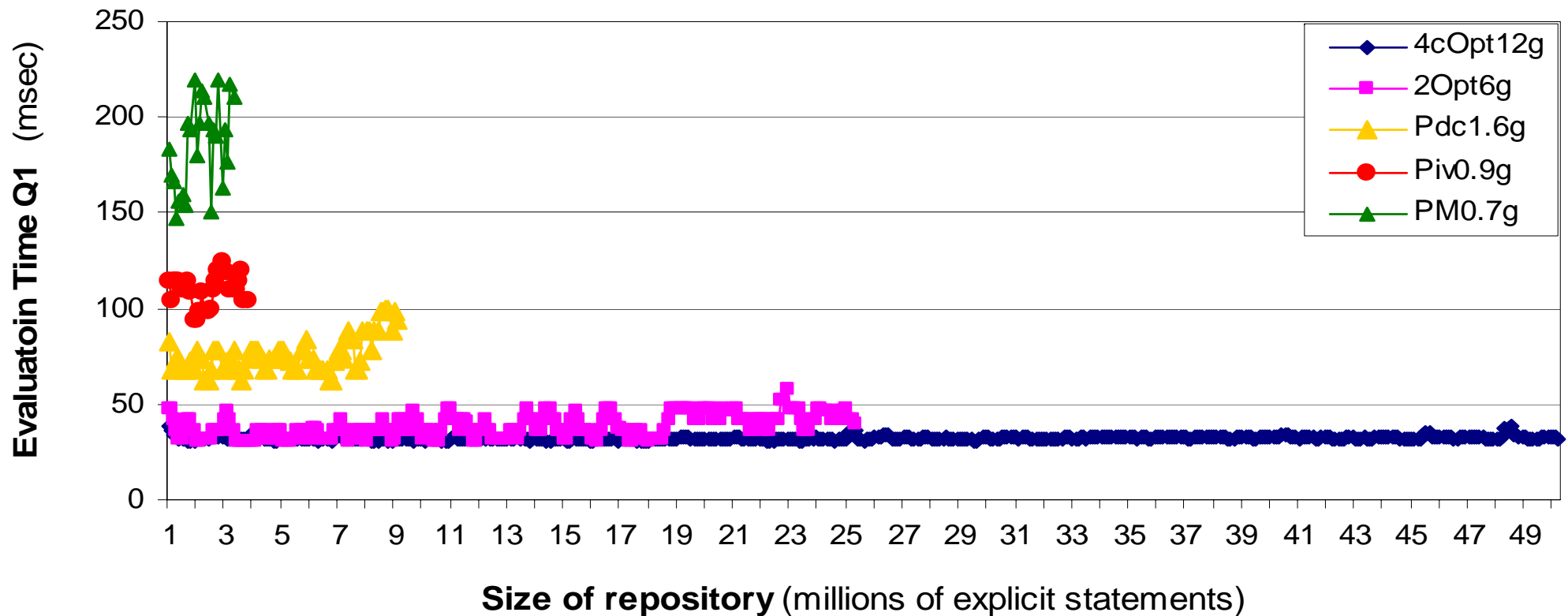
OWLIM Performance: Upload and Inference



City Benchmark: Scale and Performance

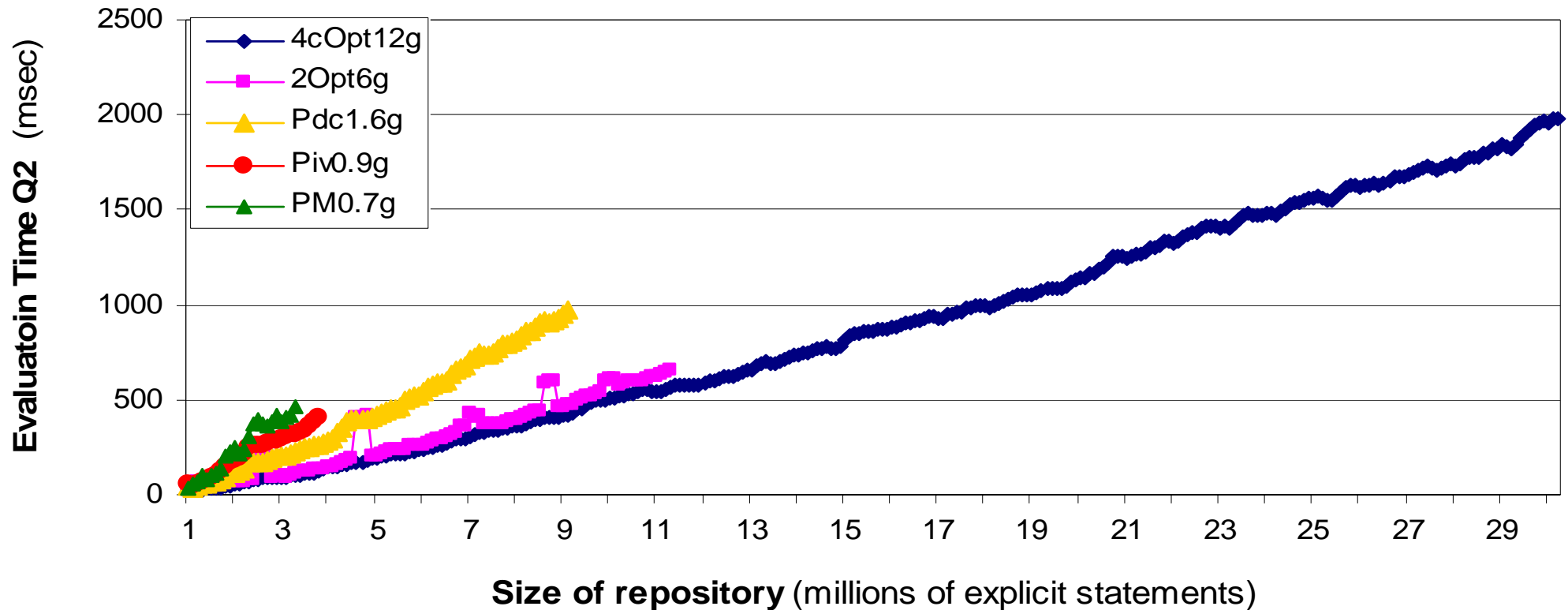
- OWLIM can manage **millions of statements** on desktop hardware:
 - Few millions of statements can be handled on a notebook model Q2'2003 with 1GB of RAM;
- Given a cheap server with 12GB of RAM (machine 4cOpt12g) it can handle **50M explicit statements**
 - As expected, the 64-bit Java VM requires a bit more memory for the same size of the repository – this explains why **2Opt6g** scales only 3 times more than **Pdc1.6g**.
- Upload speed (including inference and storage) **20,000-150,000 st./sec.**
 - It slows down in a “slow” linear dependency to the size of the repository
 - 4cOpt12g maintains speeds above 100,000 st./sec for repositories smaller than 10 million statements; it speeds down to 20 000 st./sec for repository with 50 million statements

OWLIM Performance: Query Answering



- Q1: Pattern of 11 statement-joins
- Fixed small result-set – retrieval time close to 0
- The query evaluation time is almost constant

OWLIM Performance: Query Answering (II)



- Q2: Pattern of 12 statement-joins and LIKE “*xyz*” literal constraint
- Large result set which grows linearly with the repository
- The query evaluation and retrieval time also grows linearly

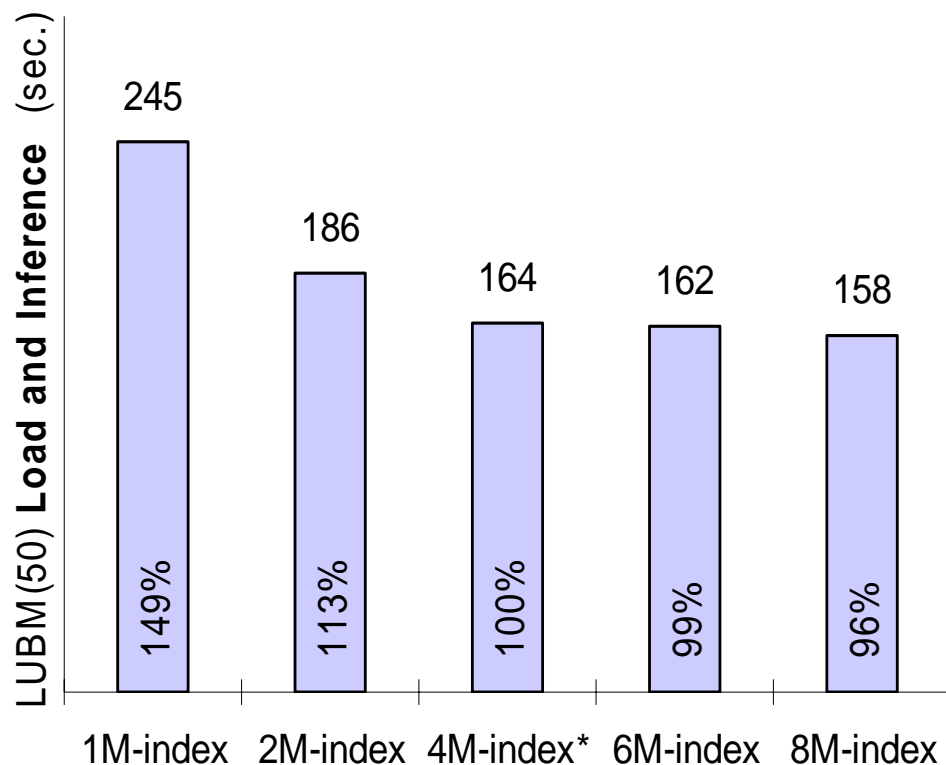
OWLIM under LUBM Benchmark

- The **Lehigh University Benchmark** (LUBM) is an outstanding repository benchmark, <http://swat.cse.lehigh.edu/projects/lubm/>
- Synthetically generated datasets on top of a **fixed OWL ontology**
- There are **14 queries**, checking different inferences and Q/A speed
- The biggest set available is LUBM(50,0) – **6.8M expl. statements**
 - About 600 MBytes in about 1000 RDF/XML files
- **OWLIM loads LUBM(50,0) in 6 min.** on a desktop machine:
 - The only other system known to load it (and answer the queries afterwards) does it in 12 hours! [Guo05]
 - All the queries are answered correctly
- OWLIM can load **LUBM(300,0)** on 2Opt6000mb machine in 50 min.
 - About 40M statements, 3GB input files, 6GB in N-Triples persistency

OWLIM Performance Analysis

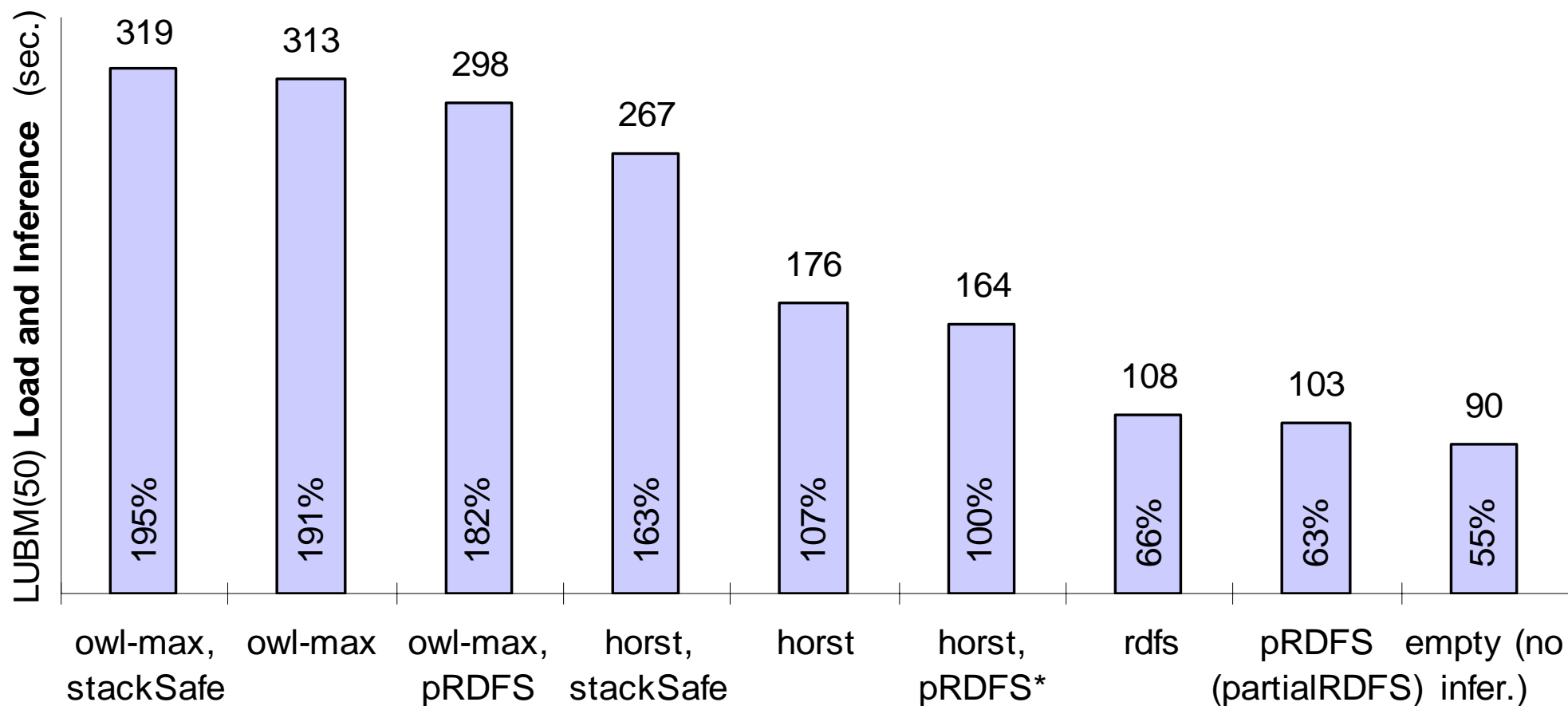
- Several tests of different configurations and parameters
- **Loading LUBM(50,0)**, which includes:
 - Parsing of the input RDF/XML files;
 - Inference – the inferred closure is calculated through forward-chaining and total materialization. This operation is not performed only when the **empty** rule-set is selected in order to force OWLIM to act as a plain RDF store;
 - Persistence (unless it is switched off) of all the data
- The **basic configuration (100%** relative score):
 - Machine 4cOpt4g: 2xOpteron 270; 12GB of RAM;
 - 64-bit JDK 1.5.0;
 - 64-bit Suse Linux, ver. 10;
 - OWLIM with its **default settings**: noPersist=false; ruleset=owl-horst; partialRDFS=true; Index-Size: 4M entries; stackSafe=false

LUBM(50,0): The Optimal Index Size Analysis



- As expected, larger index sizes lead to better performance.
- Critical for the performance on LUBM(50,0) is the border line between 1 and 2 millions of index entries.
- Index sizes larger than the default setting (4 million entries, 64MB of RAM) seem to deliver very little improvement.

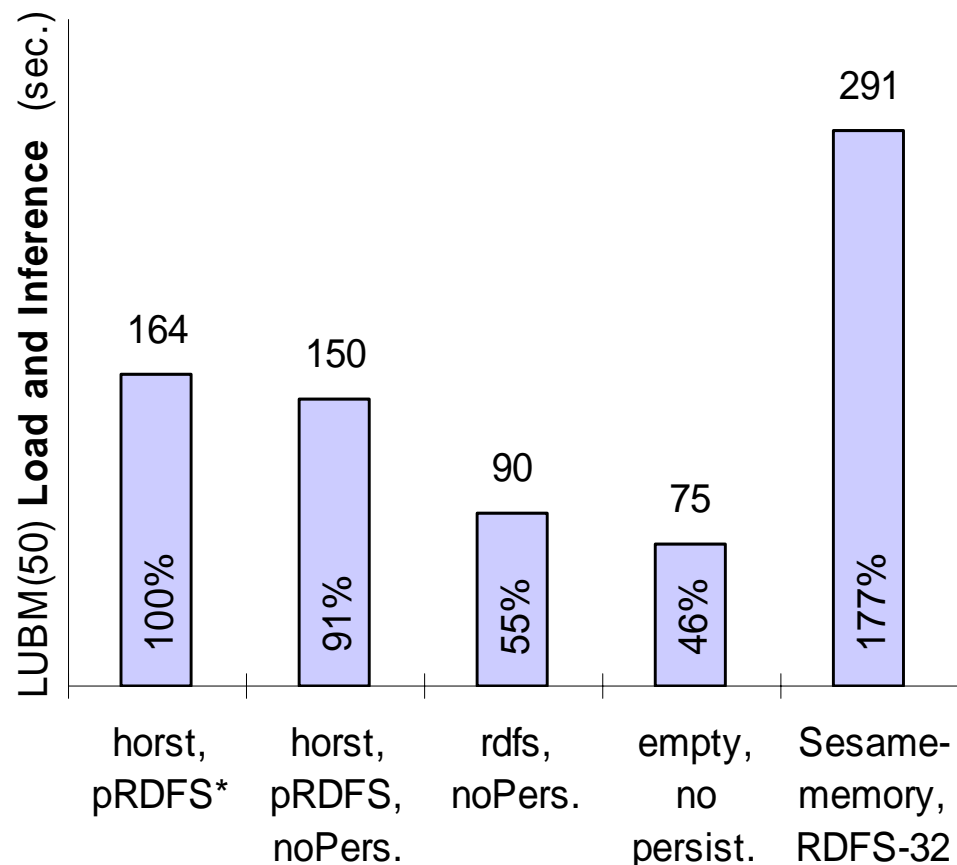
LUBM(50,0): Rule-set and Inference Mode



LUBM(50,0): Rule-set and Inference Mode (II)

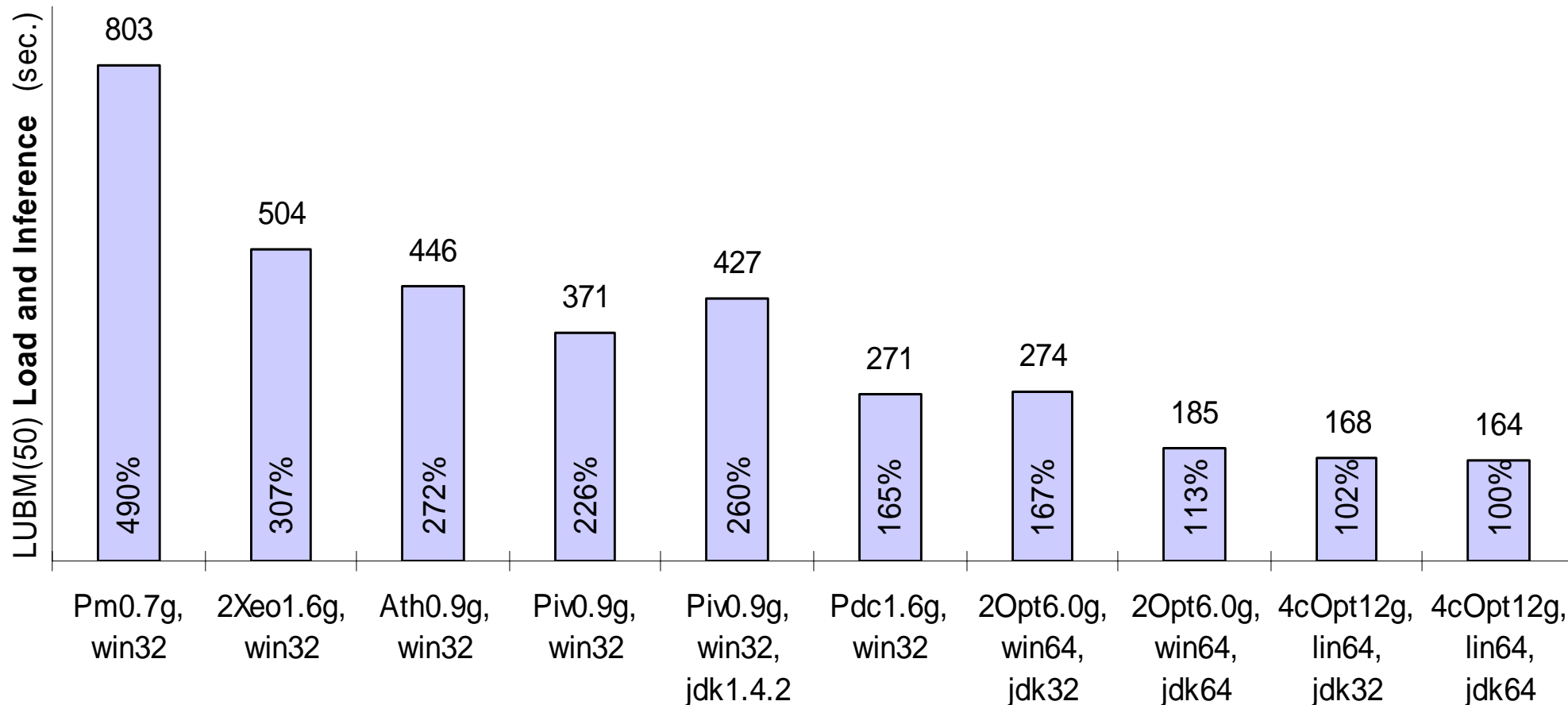
- The **partRDFS** optimization provides 20% speedup for rule set **owl-horst**, 14% for **owl-max**.
 - But it makes the basic RDFS entailment in OWLIM (rule-set **rdfs**) only about 2% faster. The conclusion is that for simpler rule-sets the **partRDFS** optimization plays smaller role.
- The inference support (for **owl-horst** with **partRDFS**) takes a bit less than half of the processing time of the default configuration
 - considering that the plain RDF version (**empty**) requires only 55% of the time to load the dataset;
- **owl-max is two times slower** than the default setup (owl-horst with partialRDFS switched on)
- The **stack-safe mode slows down by 50%** the version owl-horst (with partialRDFS switched off)
 - but has almost no impact on the owl-max inference.

LUBM(50,0): The Impact of the Persistence



- The parsing and building of in-memory representation takes about half of the load time (see "empty, noPers.");
- The persistence takes 8-10% on top of the time for loading (see "horst, pRDFS, noPers.");
 - Or flat 15 sec., 20% on top of the time for parsing;
- "Sesame-memory" is more than three times slower than the "rdfs, noPers." setup of OWLIM.
 - It performs faster on 32-bit JDK 1.5 (291 s.); the time on 64-bit JDK was even higher (329 s.).

LUBM(50,0): Different Hardware, OS, JDK



Refer to OWLIM's system documentation for analysis and comments.

OWLIM under eLUBM Benchmark

- The eLUBM, [Ma et al, 2006], is a further development of the LUBM (<http://www.alphaworks.ibm.com/tech/semanticstk>).
- eLUBM **uses the LUBM evaluation framework**, but provides alternative ontology, KB and the queries which allow for:
 - More comprehensive coverage of the **OWL Lite and DL semantics**;
 - Additional connections across the datasets for the universities;
- Two distinct datasets for OWL Lite and OWL DL, including collections for 1, 5 and 10 universities: **Lite-1, Lite-5, Lite-10, DL-1, DL-5, and DL-10**.
- **Two sets of queries** are provided – one for each of the datasets – covering various aspects of the language semantics
- [Ma et al, 2006] reports evaluation of DLDB-OWL, OWLIM (a previous version) and MINERVA:
 - OWLIM is the fastest to load Lite-1, but fails on the other datasets
 - The problem with OWLIM v.2.8.2 was **stack overflow**; this problem is solved after v.2.8.3 with the **stackSafe** parameter.

OWLIM under eLUBM Benchmark (II)

- To pass eLUBM OWLIM has to be configured as follows:
 - stackSafe=true; ruleset=owl-max
- Given 512MB of RAM on a desktop machine (Piv0.9g) OWLIM shows the following results:

	Lite-1		Lite-5		DL-1		DL-5	
Loading and inference	24 sec.		123 sec.		45 sec.		192 sec.	
Query Evaluation	Time (msec.)	Res. #	Time (msec.)	Res. #	Time (msec.)	Res. #	Time (msec.)	Res. #
Query 4	593	397	3,099	397	598	414	3,199	414
Query 11	950	1,409	19,721	6,140	1,310	1,499	28,104	6,230
Query 12	118	69	545	69	126	37	578	37
Query 14					62	6,696	242	6,696

- Results for queries evaluated in less than 100 ms (for 1 university) and under 200 ms. (for 5 universities are not shown)

BigOWLIM

- **Fully functional pre-release** of BigOWLIM is already available:
 - Check <http://www.ontotext.com/owlim/big/>
- BigOWLIM is an even more scalable **not-in-memory** version, based on the corresponding version of the TRREE engine
 - The “standard” OWLIM version, which uses in-memory reasoning and query evaluation is referred as **SwiftOWLIM**
- BigOWLIM does not need to maintain all the contents of the repository in the main memory in order to operate
- BigOWLIM stores the contents of the repository (including the “inferred closure”) in binary files; not in N-Triples
 - This allows **instance startup and initialization** of large repositories, because it does not need to parse, re-load and re-infer all the knowledge from scratch

BigOWLIM vs. SwiftOWLIM

- BigOWLIM uses **sorted indices**
 - While the indices of SwiftOWLIM are essentially hash-tables
 - In addition to this BigOWLIM maintains data statistics, to allow ...
- Database-like **query optimizations**
 - Re-ordering of the constraints in the query has no impact on the execution time
 - Combined with other optimizations, this feature delivers dramatic improvements to the evaluation time of “heavy” queries
- Special handling of **equivalence classes** (owl:sameAs)
 - Large equivalent classes does not cause excessive generation of inferred statements

BigOWLIM's Memory Usage

- The caching of BigOWLIM is highly configurable
- An XLS “calculator” is provided; follow sample configurations

		CFG1, LUBM(50,0)		CFG2, LUBM(8000,0)	
Parameter	Factor	Value	Memory Size (mb)	Value	Memory Size (mb)
key.index.size	16	500,000	8	25,000,000	381
cache-size	29	500,000	14	60,000,000	1,659
entity-index-size	4	1,000,000	4	120,000,000	458
page-cache	16,000	1,000	15	10,000	153
<i>total cache</i>			<i>41</i>		<i>2,651</i>
entity dictionary	29	1,384,243	38	221,478,842	6,125
literals	29	461,414	13	73,826,281	2,042
<i>dict+literals</i>		<i>1,845,657</i>	<i>51</i>	<i>295,305,122</i>	<i>8,167</i>
<i>trree index ref</i>	<i>32</i>	<i>2,000,000</i>	<i>61</i>	<i>2,000,000</i>	<i>61</i>
Total (MB)			153		10,818

BigOWLIM: 100M Statements on a Desktop

- It handles **LUBM(1000,0) on desktop machine** with 2GB of RAM
 - Hardware: Piv0.9g (Pentium 4, 3.0GHz, #630)
 - 32-bit JDK 1.5 given -Xmx1600
 - **Loading, inference, and storage takes 11h 20 min.**
 - LUBM(1000,0) contains above **130M explicit statements**
 - 10GB RDF/XML files; the pure parsing time is about 4h
- **LUBM(50,0) processed with only 192MB** of RAM
 - Piv0.9g; 32-bit JDK 1.5 given -Xmx1600
 - **Loading, inference, and storage takes 26 min**
 - It is only 4 times slower than the in-memory version
 - Upload speed around **4,000 st./sec.**

Reasoning over 1 Billion Statements

- BigOWLIM successfully passed **LUBM(8000,0)**
 - Hardware: 4cOpt12g (2 x Opteron 270, 16GB of RAM, RAID 10)
 - OS: Suse 10.0 Linux, x86_64, Kernel 2.6.13-15-smp
 - 64-bit JDK 1.5 given -Xmx12000
 - **Loading, inference, and storage took 69 hours** and 51 min
 - LUBM(8000,0) contains **1.06 Billions of explicit statements**
 - The “inferred closure” contains about 786M statements
 - **Managing over 1.85 Billions of statements in total**
 - 92GB RDF/XML files; 95 GB binary storage files
 - **Average Speed: 4 538 statements/sec.**

Reasoning over 1 Billion Statements (II)

Query no	Evaluation time (msec.)	Results Count
query1	1 829	4
query2	966 495	2 528
query3	15 688	6
query4	77 685	34
query5	20 663	719
query6	3 033 415	83 557 706
query7	65 401	67
query8	25 379	7 790
query9	4 736 695	2 178 420
query10	26 212	4
query11	1 186	224
query12	2 490	15
query13	321 145	37 118
query14	2 091 983	63 400 587

Thank You!

<http://www.ontotext.com/owlim>

Based on the limited evaluation results, publicly available:

**OWLIM is the fastest and most scalable
OWL semantic repository in the world!**