USING VIRTUOSO AS AN ALTERNATE TRIPLE STORE FOR A VIVO INSTANCE

PAUL ALBERT, ELIZA CHAN, PRAKASH ADEKKANATTU
AND MOHAMMAD MANSOUR

AUGUST 14, 2015

USING VIRTUOSO AS AN ALTERNATE TRIPLE STORE FOR A VIVO INSTANCE

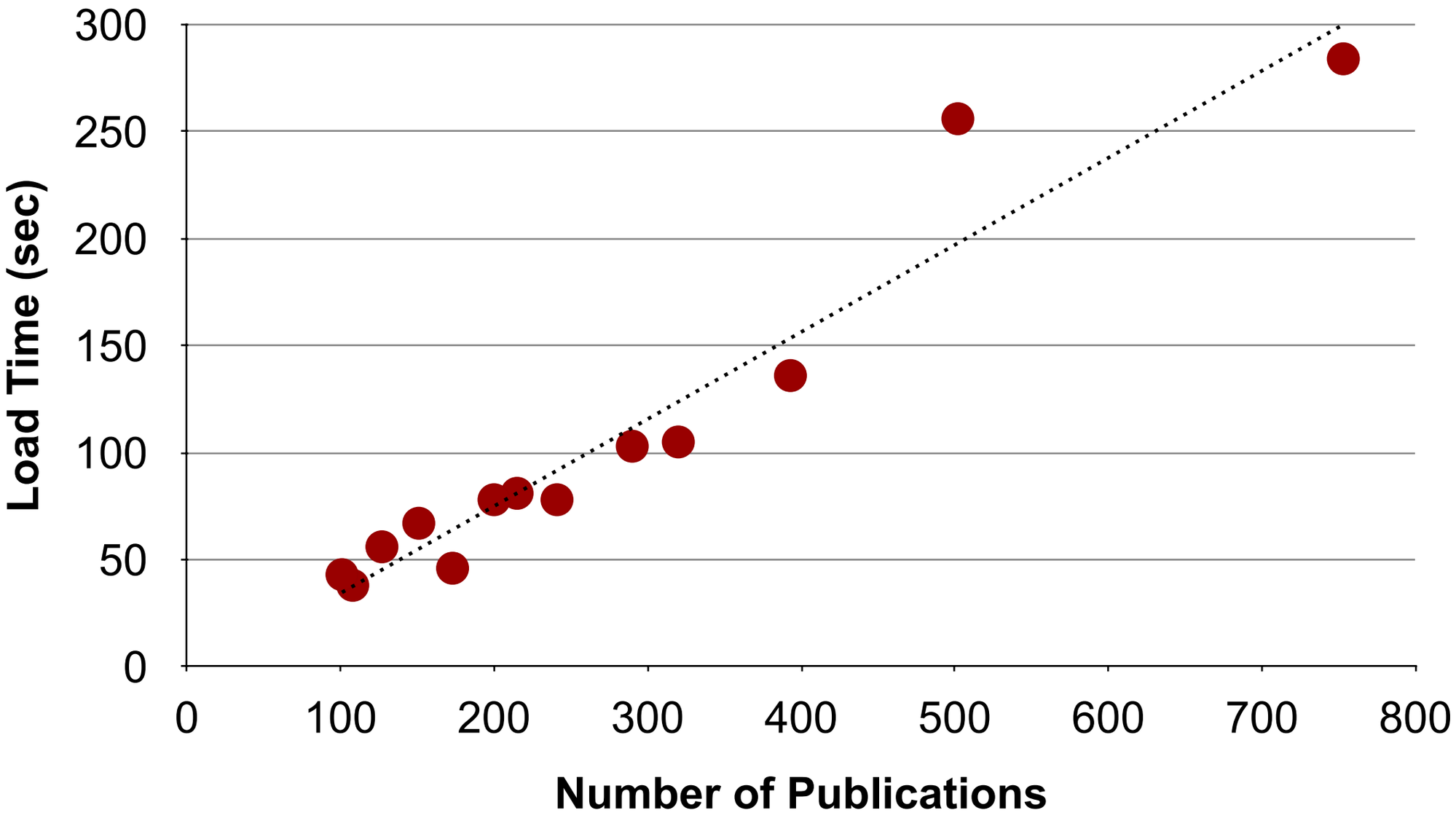
PAUL ALBERT, ELIZA CHAN, PRAKASH ADEKKANATTU
AND MOHAMMAD MANSOUR

AUGUST 14, 2015

SPECIAL THANKS TO…

- Jim Blake
- Jon Corson-Rikert
- Brian Lowe
- Brian Caruso
- Ted Lawless
- …and everyone else I harassed about performance

PROFILE LOAD TIMES c. JAN. 2013



HOW THIS AFFECTED US

- Increased downtime
- Slow loading pages
- VIVO Dashboard on hold
- Infrequent ingest
- Delayed marketing effort
- Opportunity cost of additional development
- Opportunity cost of responding to tickets and complaints
- Opportunity cost of attending meetings

ARTIST'S RENDERING OF LEADERSHIP'S REACTION TO POOR PERFORMANCE



LOCAL DB EXPERT: WHY ARE THERE 300+ QUERIES PER PAGE?

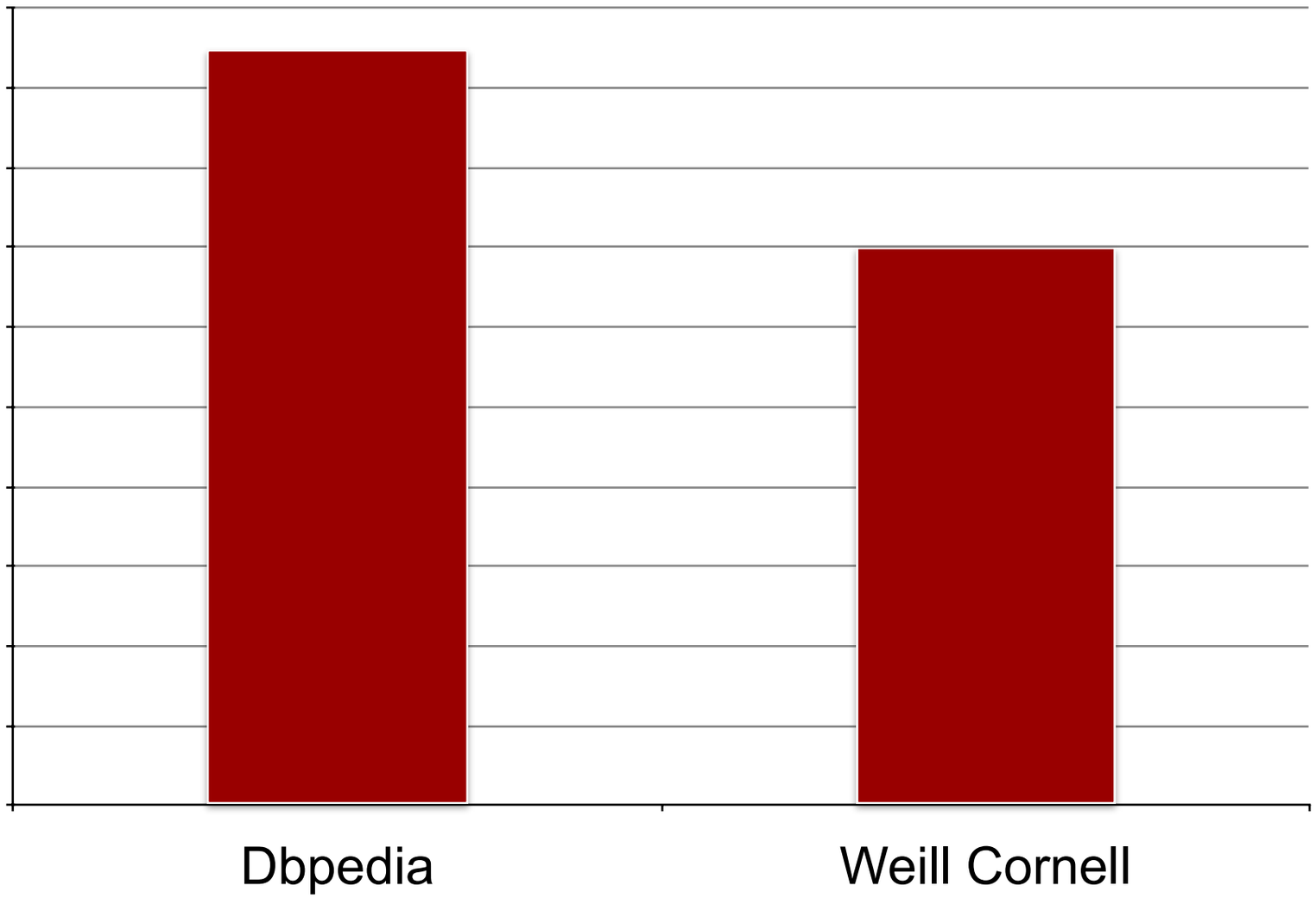
1.0	0.00%	quads - SELECT		0.793 s
3.0	0.00%	ComboPooled DataSource.get Connection()		0.794 s
1.0	0.00%	quads - SELECT		0.797 s
1.0	0.00%	ComboPooled DataSource.get Connection()		0.798 s
1.0	0.00%	quads - SELECT		0.799 s
2.0	0.00%	ComboPooled DataSource.get Connection()		0.800 s
3.0	0.00%	quads - SELECT		0.802 s
2.0	0.00%	ComboPooled DataSource.get		0.805 s
4.0	0.00%	quads - SELECT		0.808 s
1.0	0.00%	ComboPooled DataSource.get Connection()		0.812 s
1.0	0.00%	quads - SELECT		0.814 s
1.0	0.00%	ComboPooled DataSource.get Connection()		0.815 s
1.0	0.00%	VARIABLE - SELECT		0.815 s
1.0	0.00%	quads - SELECT		0.816 s
3.0	0.00%	ComboPooled DataSource.get Connection()		0.817 s
1.0	0.00%	quads - SELECT		0.820 s
2.0	0.00%	ComboPooled		0.821 s



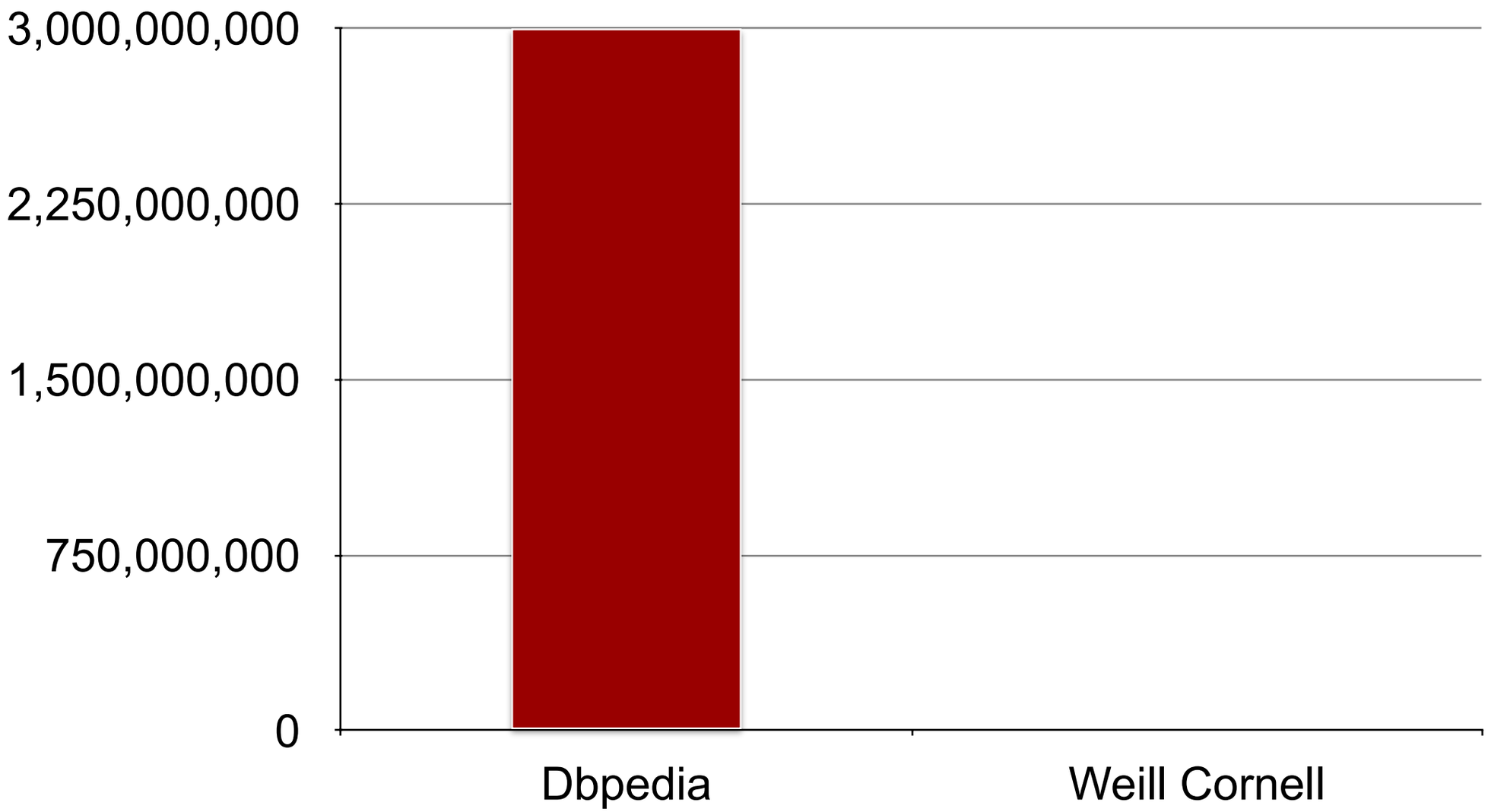
AVENUES PURSUED/CONSIDERED

- increase memory on server
- dedicated server
- increase heap size allocated to VIVO JVM
- increase cores (CPU's) on server
- reboot Tomcat nightly
- tuning Tomcat and MySQL
- moving to a high performance MySQL cluster
- throttle local and external bot access (Yahoo Slurp)
- caching
- updates to Java, Tomcat, and Apache
- reducing fanciness done in templates (e.g., no display of co-authors)
- query optimization in templates
- adjustments to restrict editing by graph policy
- lazy loading of publications
- removing data
- load balancing

DOES WCMC HAVE A LOT OF TRIPLES?



10 MILLION TRIPLES IS *NOT* THAT MANY

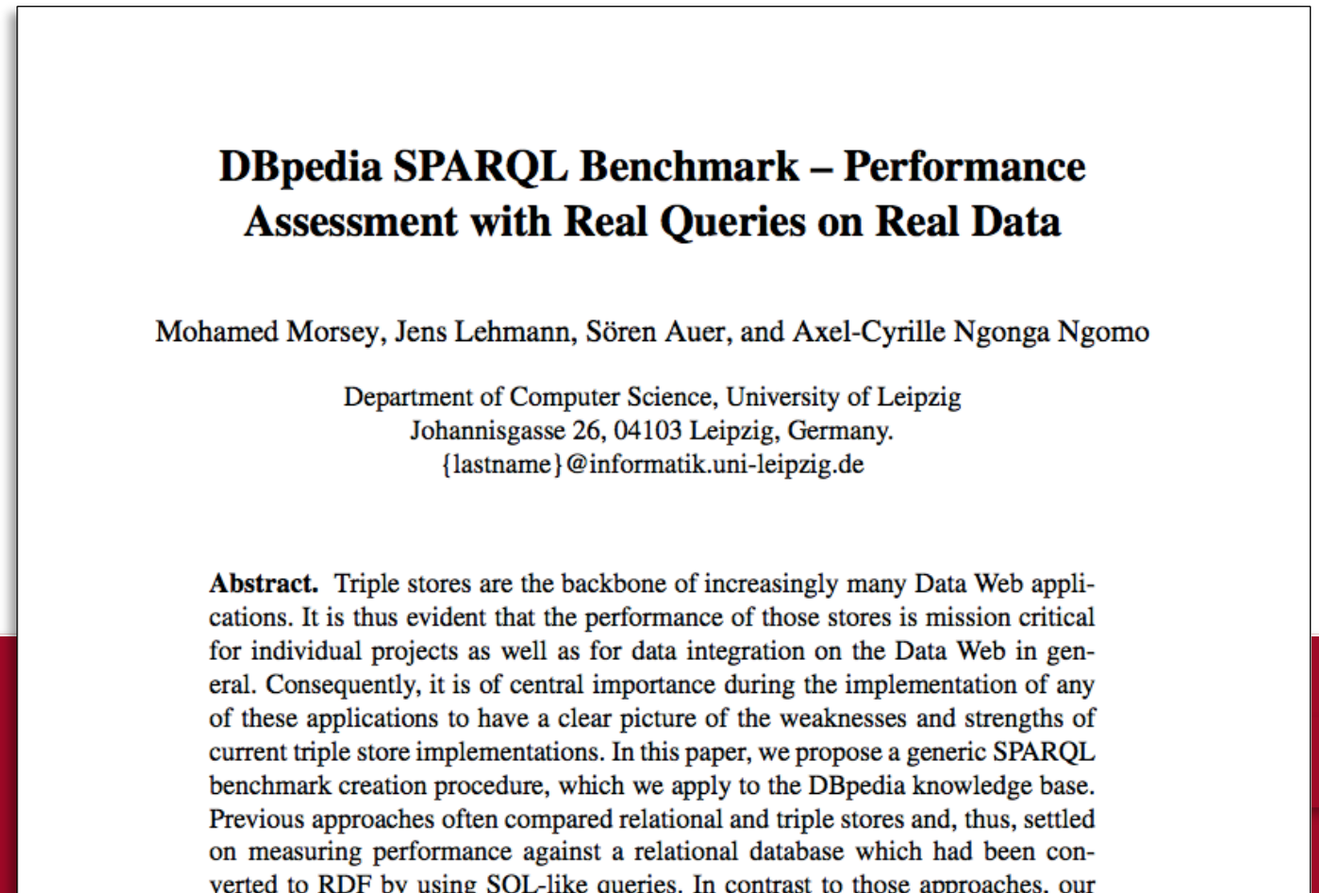


CRITERIA FOR ASSESSING TRIPLE STORES

- Is the triple store being used in production by **other VIVO sites**?
- Related: which triple stores are being used in production by **large semantic web applications**?
- Does the tool come with a **SPARQL endpoint**?
- Does the company which produces the triple store have a strong track record of **development**?
- Does the company which produces the triple store have a strong track record of **support**?
- What are the documented **query performance metrics** of the triple store against competing triple stores?

DBPEDIA SPARQL BENCHMARK

- Published in 2011
- Based on queries that were actually issued by humans and applications against existing RDF data not resembling a relational schema. Procedure for benchmark creation is based on query-log mining, clustering and SPARQL feature analysis
- Compared: Virtuoso 7, Sesame, Jena-TDB, and BigOWLIM
- Full text: http://goo.gl/PqUZM7



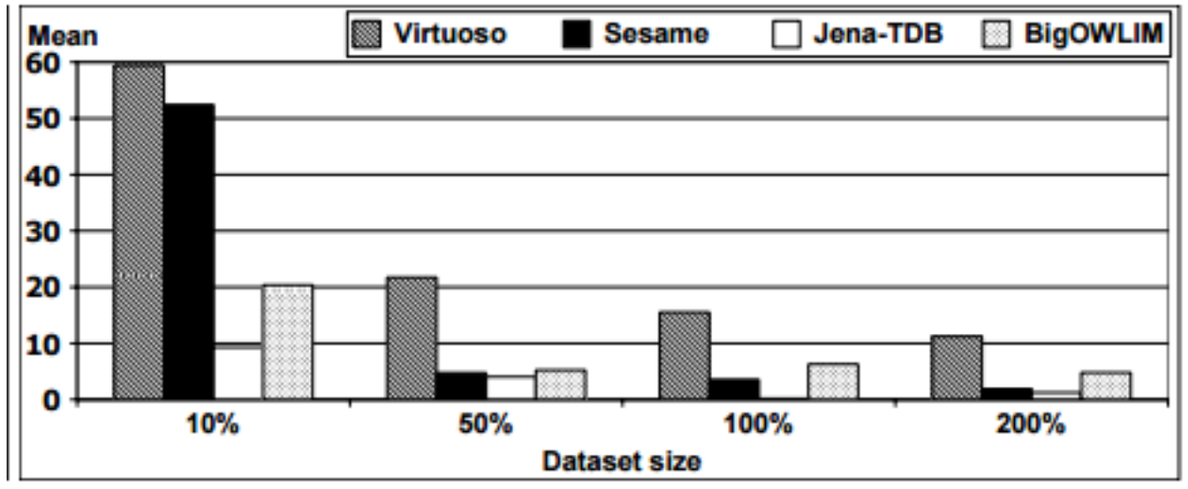
DBpedia SPARQL Benchmark – Performance Assessment with Real Queries on Real Data

Mohamed Morsey, Jens Lehmann, Sören Auer, and Axel-Cyrille Ngonga Ngomo

Department of Computer Science, University of Leipzig
Johannisgasse 26, 04103 Leipzig, Germany.
{lastname}@informatik.uni-leipzig.de

Abstract. Triple stores are the backbone of increasingly many Data Web applications. It is thus evident that the performance of those stores is mission critical for individual projects as well as for data integration on the Data Web in general. Consequently, it is of central importance during the implementation of any of these applications to have a clear picture of the weaknesses and strengths of current triple store implementations. In this paper, we propose a generic SPARQL benchmark creation procedure, which we apply to the DBpedia knowledge base. Previous approaches often compared relational and triple stores and, thus, settled on measuring performance against a relational database which had been converted to RDF by using SQL-like queries. In contrast to those approaches, our

DBPEDIA SPARQL BENCHMARK: RESULTS



Geometric mean of queries per second

Conclusions: Virtuoso leads the field with a substantial head start of double the performance for the 10% dataset (and even quadruple for other dataset sizes) compared to the second best system… Jena-TDB can in general not deliver competitive performance with being by a factor 30-50 slower than the fastest system….

BERLIN SPARQL BENCHMARK (BSBM)

- Published in 2013
- Hypothetical use cases
- Compared: Virtuoso 7 and 6, BigData, Jena-TDB, and BigOWLIM
- Full text: http://goo.gl/3GM2pM

BERLIN SPARQL BENCHMARK (BSBM): RESULTS

Dataset Size 10M		Number of clients		
	1	4	8	64
BigData	7.290	16.222	16.439	18.812
BigOwlim	121.841	265.294	177.338	218.678
TDB	7.468	17.698	9.503	8.414
Virtuoso6	**431.465**	**2667.657**	**3915.854**	**1401.186**

Dataset Size 100M		Number of clients		
	1	4	8	64
BigData	12512.278	17949.632	19574.007	20422.626
BigOwlim	14029.453	17184.314	11677.860	8321.202
TDB	15381.857	19036.097	24646.705	14838.483
Virtuoso6	37678.319	64885.747	112388.811	20647.413
Virtuoso7	**47178.820**	**91505.200**	**188632.144**	**216118.852**

Query Mixes per Hour for Multiple Clients

1. BI use case
2. Explore use case

	10M	100M	1B
BigData	7.290	-	-
BigOwlim	121.841	15.512	1.400
TDB	7.468	-	-
Virtuoso6	**431.465**	35.342	2.383
Virtuoso7	-	**996.795**	**75.236**

Query Mixes per Hour for Single Clients

3. Explore use case
4. BI use case

	100M	200M	1B
BigData	12512.278	10059.940	-
BigOwlim	14029.453	9170.083	1669.899
TDB	15381.857	10573.858	-
Virtuoso6	37678.319	**32969.006**	8984.789
Virtuoso7	**47178.820**	-	**27933.682**

VIRTUOSO VS. JENA

System	Jena	Virtuoso
Sustainabiltiy	Apache foundation (non-profit)	OpenLink Software
Major releases	RDB, SDB (ships with VIVO 1.7.*), TDB	6 and 7
Storage	MySQL database (RDB, SDB); native triple store (TDB)	native RDF database
SPARQL endpoint	needs to be configured	standard
License / Sustainability model	Apache License, Version 2.0 / Apache Foundation	- OpenLink Virtuoso (GNU General Public License v2): no cluster store - Virtuoso Universal Server (commercial version): has cluster store
Support	Listservs	Listservs and paid support

JENA SDB DATA STORE (QUADS)

g	s	p	o
4842840998053235027	-9223344952870381322	-6430697865200335348	-7985466041922445122
786347385076487474	-9223344952870381322	-6430697865200335348	-2319456160028125983
786347385076487474	-9223344952870381322	-6430697865200335348	8662362958342704019
786347385076487474	-9223344952870381322	-3346177138706421939	-7985466041922445122
4842840998053235027	-9223344952870381322	-1057367022653961137	8962174421294010694
4842840998053235027	-9223344952870381322	3709441663267393283	4398950194442419917
4842840998053235027	-9223344952870381322	4270502025570551231	2154719398351543909

Content

Column Name	Type	Length	Flags	Null	De
g	bigint	20	u z	NO	
s	bigint	20	u z	NO	
p	bigint	20	u z	NO	
o	bigint	20	u z	NO	

Columns

Indexes

Index Name	Kind	Type	Columns
SubjPredObj	INDEX	BTREE	s p o
PredObjSubj	INDEX	BTREE	p o s
GraPredObj	INDEX	BTREE	g p o
GraObjSubj	INDEX	BTREE	g o s
ObjSubjPred	INDEX	BTREE	o s p

+ −

Primary Key

Column	Length	Auto Increment
g		Make
s		Make
p		Make
o		Make

Indexes and keys

JENA SDB DATA STORE (NODES)

Content

hash	lex	lang	datatype	type
-9223344952870381322	http://vivo.med.cornell.edu/individual/pubid69			2
-9223342742734684175	In-Gel Stable-Isotope Labeling (ISIL): a strategy			3
-9223339030457833622	http://vivo.med.cornell.edu/individual/citation-			2
-9223338740967586793	Printing		http://www.w3.org/2001/XMLSchema#string	4
-9223332137690944932	Weinstein MW			3
-9223316123889510431	Cataruozolo			3
-9223312659079146459	84889071052			3
-9223307283356944572	Authorship for Taghavi P			
-9223289730646898752	http://vivo.med.cornell.edu/individual/pubid00			
-9223281446583960956	http://vivo.med.cornell.edu/individual/pubid77			
-9223280376072251202	http://vivo.med.cornell.edu/individual/pubid00			

Indexes and keys

Indexes

Index Name	Kind	Type	Columns

+ −

Primary Key

Column	Length	Auto Increment
hash		Make
lex		
lang		
datatype		
type		Make

Columns

Column Name	Type	Length	Flags	Null	Default
hash	bigint	20	u z	NO	0
lex	longtext			YES	NULL
lang	varchar	10		NO	
datatype	varchar	200		NO	
type	int	10	u z	NO	0

VIRTUOSO'S BACK END

549 MB virtuoso.db vs. 1.4 GB MySQL dump

```
$ tail -1 virtuoso.db
?}?}?}?}?}?}e[e[?}?}??}?}?}e[e[??}?}e[??}?}??e[??}?H}?}e[1??}?\e[??}?}?3e[?H}??
<?F??U?}?}
e[!"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
""""""""""""""""""!"""!"""""""""""""""""""""""""""""""""""""""""""""""""""""""""
""""""""""???}?}
e["""""""""""""""""""""""""!"""""""""""""""""""""""""""""""""""""""""""""??}?}?
e[??}?e[??}?}?[?:}?<???}?}?e[??}?}??e[????}?}
e[!"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
""""""""""""""""""""""""""""""""""!"""""""""""""""!""""""!""""""""""""""""""""""
""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""???}??e[?>?<???}?}?
e[??}?}
e["""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
""""""""""""""""""""""""""!""""""""""""""""""""""""""""""""""!""""""""""""""""""
""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
""""""""""""""""""""""""""??>?}?}
e[!"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""!""""
""""""""""""""""""""""!""!"""""""""""""""""""""""""""""""""""""""""""""""""???}?
}e[!"""""""""""""""""""""""!?'<?}?}??u?}?}?~??}?}□□}??b?}?}?)@?}?}??&?}?}??
B?}?}????}?}?ă}?,E?}?}?]??}?}?\??}?}????}?}????}?}????}?}????}?}?`??}?}?k??}?}?
m??}?}?S??}?}e[!"""""""""""""""""""""""""""??e[?4E?}?}e[e[ ?
```

EARLY INDICATIONS OF PERFORMANCE

Criterion	Jena	Virtuoso	Improvement
Administrator load of publications graph	2 hours	10 seconds	720x
Sophisticated queries run against SPARQL endpoint	20 seconds	2 seconds	10x
Backup	1.5 hours	1 minute	90x
Load of individual profile	30 seconds	20 seconds	0.5x

Virtuoso SPARQL Query Editor

About | Namespace Prefixes | Inference rules

Default Data Set Name (Graph IRI)

Query Text

```
PREFIX wcmc: <http://weill.cornell.edu/vivo/ontology/wcmc#>
SELECT DISTINCT ?doc ?doctype ?title ?volume ?pmid ?doi ?scopusDocId ?pStart ?pEnd ?dt ?
journal
WHERE {
?doc rdf:type bibo:Document .
?doc rdf:type ?doctype .
?doc rdfs:label ?title .
?doc bibo:volume ?volume .
?doc vivo:hasPublicationVenue ?journal .
?doc vivo:dateTimeValue ?dtv .
?dtv vivo:dateTime ?dt .
optional { ?doc bibo:pageStart ?pStart }
optional { ?doc bibo:pageEnd ?pEnd }
optional { ?doc bibo:pmid ?pmid }
optional { ?doc bibo:doi ?doi }
optional { ?doc wcmc:scopusDocId ?scopusDocId }
FILTER(regex(str(?dtv), "2013")) .
}
```

Sponging: Use only local data (including data retrieved before), but do not retrieve more

Results Format: CSV

Execution timeout: 0 milliseconds *(values less than 1000 are ignored)*

Options: Strict checking of void variables

(The result can only be sent back to browser, not saved on the server, see details)

Run Query Reset

Copyright © 2015 OpenLink Software
Virtuoso version 07.20.3212 on Linux (x86_64-unknown-linux-gnu), Single Server Edition

VIRTUOSO IS GENERALLY FAST, BUT QUERIES REQUIRED TO LOAD A PAGE ONLY HAVE A 50% IMPROVEMENT??

```
...
  UNION {
<http://mydomain.edu/individual/userid> <http://vivoweb.org/ontology/core#relatedBy> ?authorship .
   ?authorship a core:Authorship .
   ?authorship ?authorshipProperty ?authorshipValue
} UNION {
   <http://mydomain.edu/individual/userid> <http://vivoweb.org/ontology/core#relatedBy> ?authorship .
   ?authorship a core:Authorship .
   ?authorship core:relates ?infoResource .
   ?infoResource a <http://purl.obolibrary.org/obo/IAO_0000030> .
} UNION {
   <http://mydomain.edu/individual/userid> <http://vivoweb.org/ontology/core#relatedBy> ?authorship .
   ?authorship a core:Authorship .
   ?authorship core:relates ?infoResource .
   ?infoResource a <http://purl.obolibrary.org/obo/IAO_0000030> .
   ?infoResource ?infoResourceProperty ?infoResourceValue
} UNION {
   <http://mydomain.edu/individual/userid> <http://vivoweb.org/ontology/core#relatedBy> ?authorship .
   ?authorship a core:Authorship .
   ?authorship core:relates ?infoResource .
   ?infoResource a <http://purl.obolibrary.org/obo/IAO_0000030> .
   ?infoResource bibo:reproducedIn ?appearsInObj .
   ?appearsInObj rdfs:label ?appearsIn
...
```

FEEDBACK FROM VIRTUOSO COMMUNITY

while i do not know the statistics of your dataset, the illustrated query is constructed to effect such a remarkably ineffective projection, that it is no surprise that it would take twenty five seconds on a 7-million statement dataset.

the successive bgp's [basic graph patterns] - to a large extent, combine a initial projection with the same projection augmented with another bindings and thereby either reduce in cardinality or increased, depending on the selectivity of the respective addition.

not only is this remarkably inefficient in solution count, but in order to produce it, the same sub-pattern is re-evaluated eight times.

then, in the end, this highly redundant solution field is that passed through the construct filter, which eliminates duplicates and discards unbound patterns.

if one is serious about production efficiency, one should seriously consider how the application could be modified to formulate a more purposeful query.

best regards, from berlin

CUSTOMIZATIONS REQUIRED TO MAKE VIVO AND VIRTUOSO PLAY NICE

- **UNION —> OPTIONAL** - Commented out the `CONSTRUCT` queries in templates (necessary for Jena SDB) and replaced them with `OPTIONAL` queries
- **Line breaks** - Virtuoso would throw an error or create duplicate entries when a field with line breaks (e.g., overview statement) was submitted
- **FileGraphSetup class has been modified** - `dbModel.removeAll()` only when the `dbModel` and `fileModel` are not isomorphic and differ in size (Improvement in start time: 2 days+ > 30 seconds)
- **Blank nodes -** Virtuoso doesn't allow using INSERT DATA to insert blank nodes but using "INSERT" to insert blank nodes is permitted. Fix: Modify the method executeUpdate to replace "INSERT DATA" with “INSERT"
- **Harvester** - Update so it can ingest to Virtuoso

ELIZA'S BRAIN DUMP ABOUT VIRTUOSO

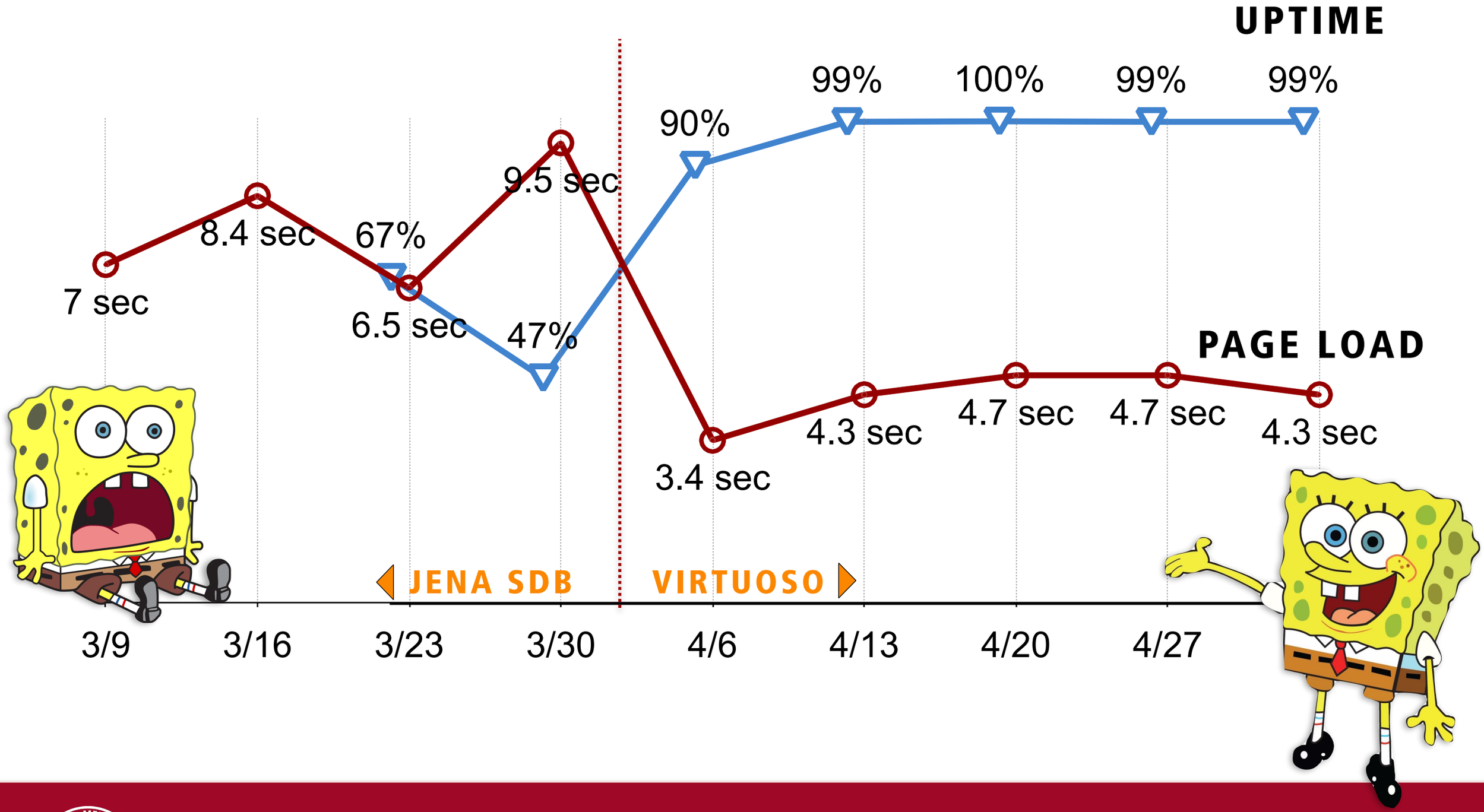
Added by Paul Albert, last edited by Paul Albert on Aug 09, 2015

Virtuoso Open Source and VIVO

- Virtuoso Open Source and VIVO
  - Download
  - Installation
  - Starting up
  - Shutdown
  - Configurations
  - Data loading
    - Step 1: Dumping data from VIVO
    - Step 2: Start with a blank Virtuoso database
    - Step 3: Load the data into Virtuoso
  - Virtuoso customization
  - Virtuoso as SPARQL endpoint for VIVO
  - Harvester
  - ./autogen.sh
  - Scheduled Backup of the Virtuoso Database
    - Schedule (vivo-prod)
    - Locations of the backup data and scripts
    - Estimated sizes of the backed up files
    - Restoring backed up data
  - Inserting Data using the SPARQL endpoint
  - Deleting Data using the SPARQL endpoint
  - VIVO customization
    - 1) FileGraphSetup class
    - 2. RDFServiceSparql class
    - 3. listViewConfig-authorInAuthorship.xml
    - 4. Known issues
    - 5. Duplications and errors when submitting the overview
      - i) RdfLiteralHash class
      - ii) RDFServiceSparql class
      - iii) EditN3GeneratorVTwo class
      - iv) SparqlQueryUtils class
  - isql Useful Tips

http://bit.ly/1WavrEN

VIVO PERFORMANCE: BEFORE & AFTER



SELECT BELLS AND WHISTLES

- Tracing - http://docs.openlinksw.com/virtuoso/fn_trace_on.html
  - For example, we learned the time it took to compile the query (36 milliseconds) and time to execute (46 milliseconds) in the engine and the remainder is server is server I/O.
- Web crawler - linkeddata.uriburner.com
  - Allows you to extract structured data from a given domain and transform that into some meaningful structure

CUSTOMIZATION OF VIRTUOSO

- Allocated 10 GB of system memory
- Default threads per query was set to 8 on commercial version whereas the OS the threads per query is 4 on the OS version. We went to 8.
- Increased max client connections to 100
- ResultSetMaxRows = 300000 (default setting makes startup fail)

KNOWN ISSUES

- Reinferencing has hiccups - Virtuoso doesn't allow deletion of blank nodes(??), which causes reinferencing problems

```
java.lang.RuntimeException:
edu.cornell.mannlib.vitro.webapp.rdfservice.RDFServiceExcept
ion: Unable to perform SPARQL UPDATE
```

- Logged in user page loads are still slower than acceptable - probably related to use of graphs

ADDITIONAL OPPORTUNITIES FOR PERFORMANCE IMPROVEMENT

49 / 100 Suggestions Summary

! Should Fix:

Enable compression

Compressing resources with gzip or deflate can reduce the number of bytes sent over the network.

Enable compression for the following resources to reduce their transfer size by 556KiB (71% reduction).

Compressing http://vivo.med.cornell.edu/vivo/js/jquery-1.9.1.js could save 184.1KiB (70% reduction).
Compressing http://vivo.med.cornell.edu/…uery-ui/js/jquery-ui-1.8.9.custom.min.js c
Compressing http://vivo.med.cornell.edu/js/tiny_mce/tiny_mce.js could save 107.3Ki
Compressing http://vivo.med.cornell.edu/js/jquery.js could save 54.3KiB (65% reduc
Compressing http://vivo.med.cornell.edu/…lugins/qtip/jquery.qtip-1.0.0-rc3.min.js co
Compressing http://vivo.med.cornell.edu/js/json2.js could save 11.8KiB (69% reduct
Compressing http://vivo.med.cornell.edu/vivo/themes/wcmc/js/individualUtils.js could
Compressing http://vivo.med.cornell.edu/js/individual/individualUriRdf.js could save
Compressing http://vivo.med.cornell.edu/js/individual/individualQtipBubble.js could s

Minify JavaScript

Compacting JavaScript code can save many bytes of data and speed up downloading, parsing, and execution time.

Minify JavaScript for the following resources to reduce their size by 155.8KiB (31% reduction).

Minifying http://vivo.med.cornell.edu/vivo/js/jquery-1.9.1.js could save 121.3KiB (47% reduction).
Minifying http://vivo.med.cornell.edu/js/json2.js could save 13.8KiB (80% reduction).
Minifying http://vivo.med.cornell.edu/…uery-ui/js/jquery-ui-1.8.9.custom.min.js could save 9KiB (5% reduction).
Minifying http://vivo.med.cornell.edu/vivo/themes/wcmc/js/individualUtils.js could save 2.7KiB (54% reduction).
Minifying http://vivo.med.cornell.edu/js/vitroUtils.js could save 2.2KiB (59% reduction).
Minifying http://vivo.med.cornell.edu/js/individual/individualQtipBubble.js could save 2.1KiB (55% reduction).
Minifying http://vivo.med.cornell.edu/js/individual/individualUriRdf.js could save 2.1KiB (52% reduction).
Minifying http://vivo.med.cornell.edu/…/individual/individualProfilePageType.js could save 1.4KiB (51% reduction).
Minifying http://vivo.med.cornell.edu/…emes/wcmc/js/plugins/jquery.truncator.js could save 1.2KiB (37% reduction).

▲Hide details

QUESTIONS?

Contact me at paa2013@med.cornell.edu.