Optimized SPARQL performance management via native API

Ontology Summit 2014 (Track-E) Hackathon, project #3 preliminary report

Contents

Project Goal

The project goal was running SPARQL queries on several triplestores to understand the advantages of one or another triplestore on one or another query, the factors affecting their performance.

Triplestore Selection

We've selected the following triplestores as performance leaders on RDF market:

- Virtuoso Universal Server Release 7.1 (http://virtuoso.openlinksw.com/)
- Stardog 2.1.2 (http://stardog.com/)
- NitrosBase RDF Storage 1.0 Release Candidate (http://nitrosbase.com/)

The selected triplestores have the following important advantages:

- Very high performance on sp2bench benchmark
- Linux and Windows versions
- Native API for fast query processing

Note: For simplicity, we used Windows versions the tools above.

Usually people use endpoints to compare triplestores. However, endpoints put a big overhead on a triplestore performance. That may influence on the performance comparison result. That's

why we used native API for our experiment. All 3 tools have necessary components to fit to our needs:

- Virtuoso has fast ADO.NET provider.
- Stardog has special fast dotnetrdf classes.
- NitrosBase RDF provides native API.NET classes.

Note: We suggested to participants to change/extend the set of triplestores, but received no proposals.

Benchmark queries

We have designed an advanced set of queries (see Appendix 2) based on recognized SP²Bench benchmark. Those queries extend of SP²Bench core query set and stress the attention on:

- search the small range of values
- search the big range of values
- Sorting
- Aggregation
- Various join queries

Note: We offered participants to change/extend the set of queries, but received no feedback.

Benchmark Application

To facilitate the setup stage and save experiment preparation time we developed the benchmark application in advance. We supposed to modify the application during the Hackathon if someone offered change in the triplestore set, but it was not needed. Basically the application opens connection to each triplestore and runs the queries on it.

Experiments

We set up experiments on 3 participants' computers. The computers have the following configurations:

Computer 1

Processor: Intel Core i5-3570 CPU @3.40 Ghz; Memory: 32 Gb; SSD: Corsair Force GS 240 Gb; Windows 8.1 x64

Computer 2

HP Compaq 8100 workstation with Intel Core i5; CPU @ 2.80 GHz, with 8Gb RAM; HD: Hitachi HDS721025CLA382 ATA (232GB)
Windows Server 2008 R2 Standard SP1.

Computer 3

Processor i5 3570, 3400MHz, 16 GB CORSAIR Vengeance CMZ8GX3M1A1600C10 DDR3 memory; OCZ Vertex 3 Max IOPS VTX3MI-25SAT3-120G 120GB SSD.
Windows 8.1 x64

We experimented with 25 mln triple datasets generated by the generation utility downloaded from SP2Bench site (http://dbis.informatik.uni-freiburg.de/content/projects/SP2B/docs/sp2b-v1_01-full.tar.gz).

Each run the query was sent to server and full result was read. That was done to avoid an idle run of query, caused by the fact that some database tools don't run query physically until the result is read.

Each query run 10 times and median value has been taken as the result value.

We've preliminarily tested the databases and have been seen that the tools we considered don't cache the result. That made us convinced that 10 times run gives us reliable median value.

All the diagrams below present the Computer 1 results. The results of Computer 2 and Computer 3 didn't show considerably different results.

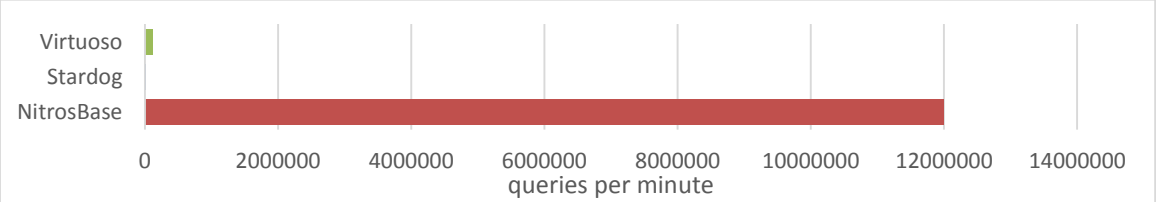
Due to limited time, we decided to run standard SP2Bench queries (Query 1 – Query 12C) on computer 1 run only. Extended queries (Query 20 – Query 64) run on all 3 computers.

The results

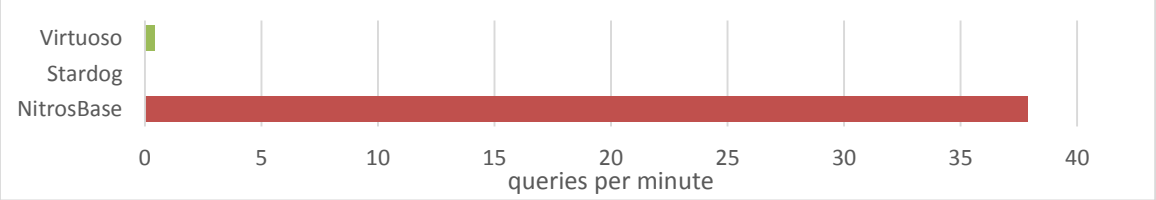
STANDARD SP2BENCH QUERIES

Larger bars indicate better performance.

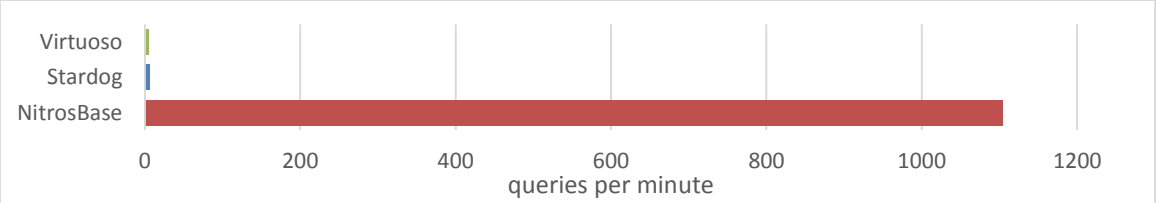
Query1



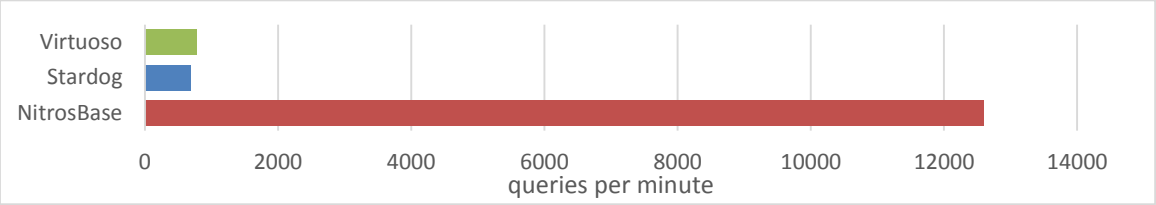
Query2



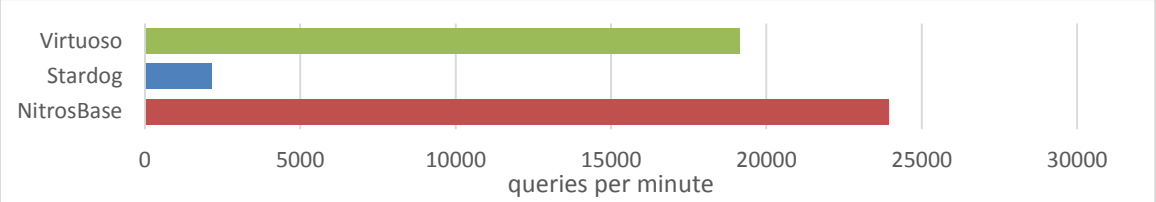
Query3a



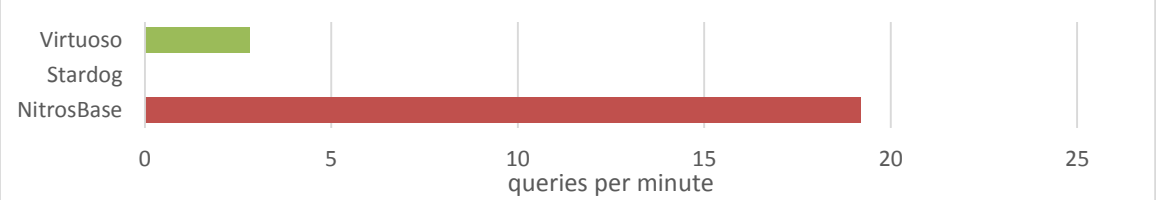
Query3b



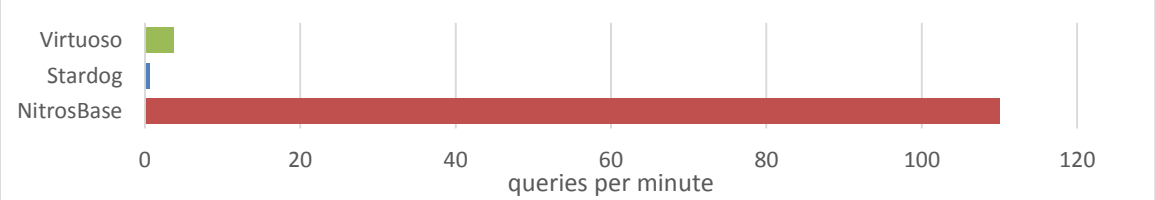
Query3c



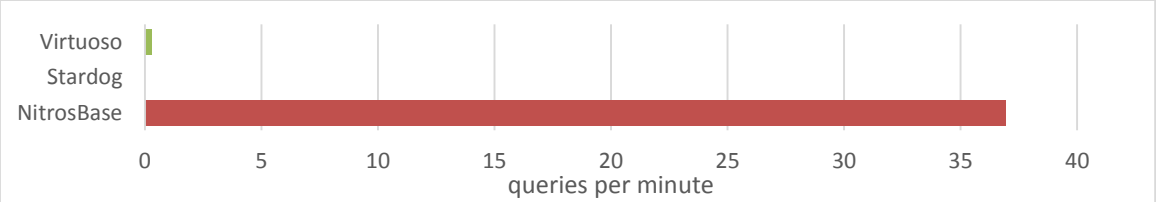
Query5a



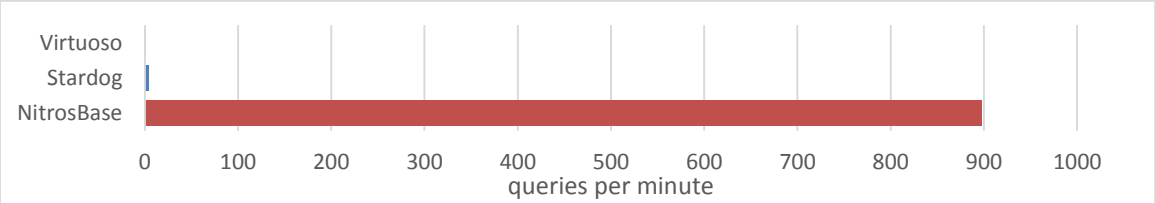
Query5b



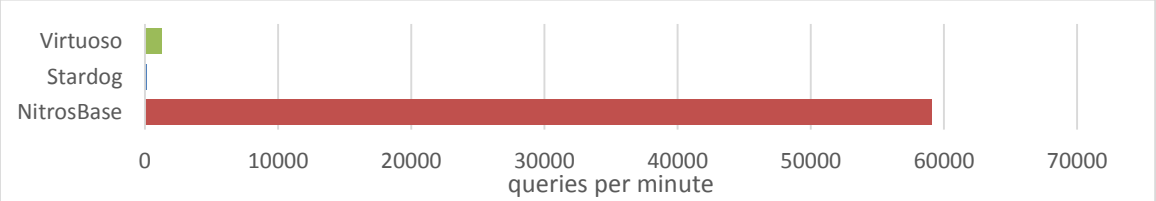
Query6



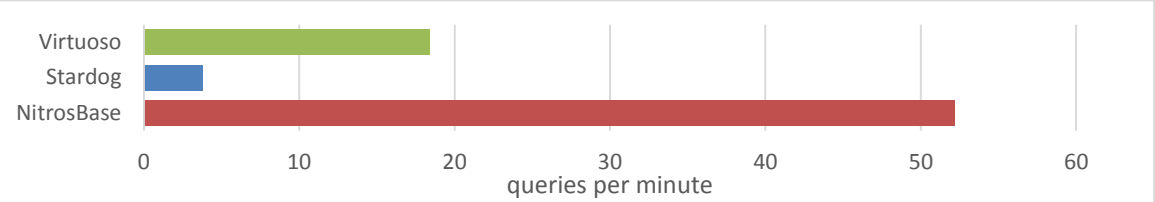
Query7



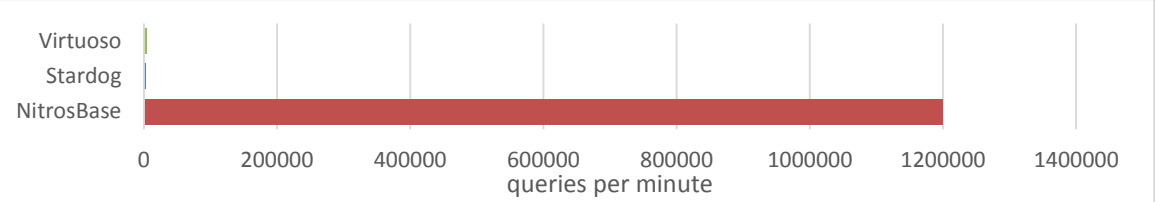
Query8



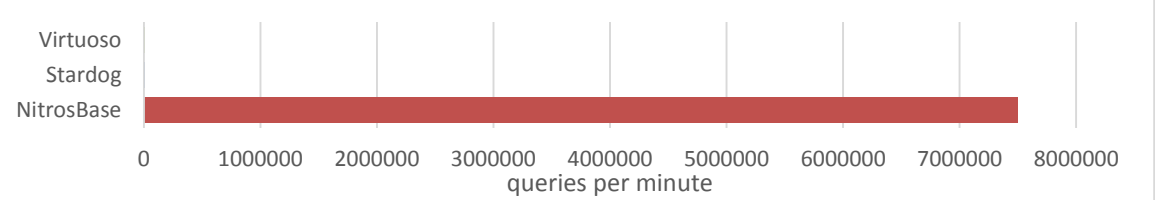
Query9



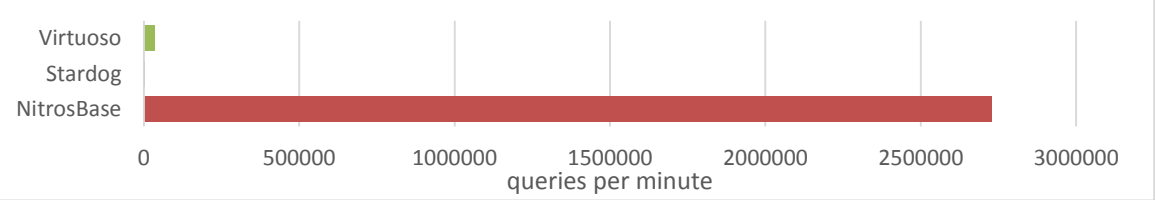
Query10



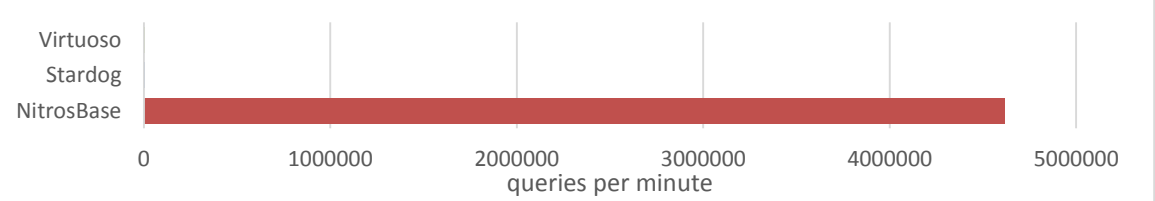
Query11



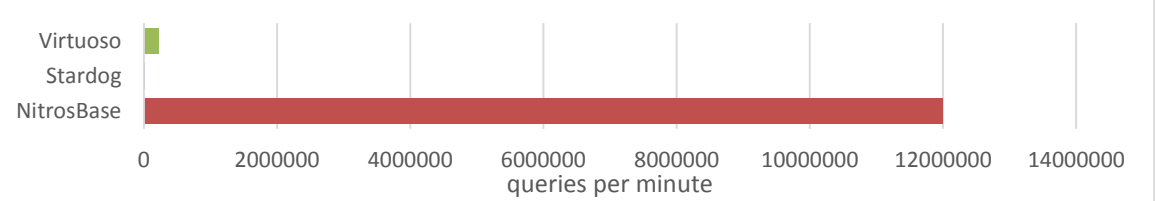
Query12a



Query12b



Query12c

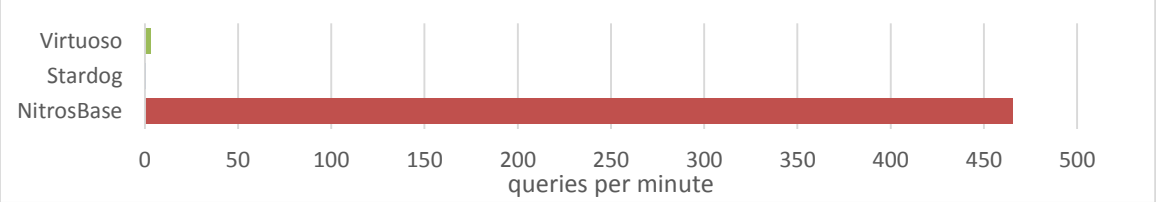


SIMPLE FILTER (ONE VARIABLE)

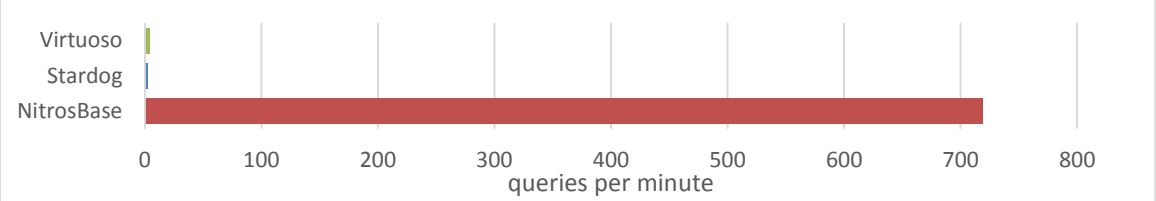
Larger bars indicate better performance.

```
select ?x ?y ?title
where{
    ?x dcterms:issued ?y.
    ?x swrc:pages ?pages.
    ?x dc:title ?title.
    filter(?y >= "2005"^^xsd:integer)
}
```

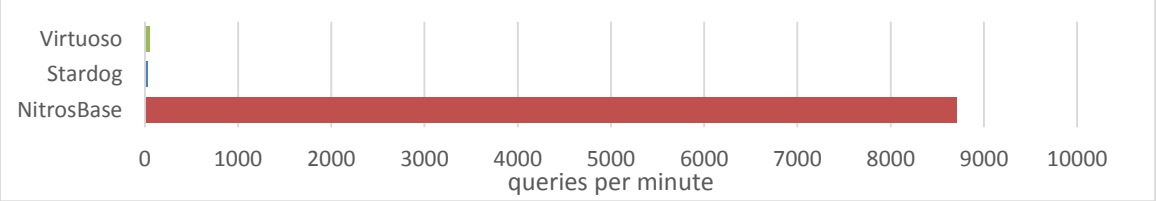
Query 20



Query 21



Query 22

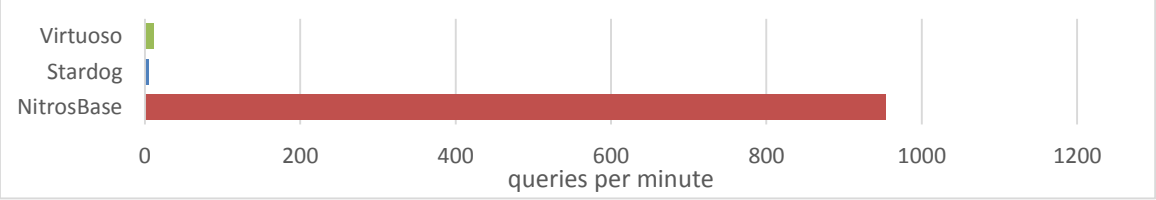


FILTER (TWO VARIABLES)

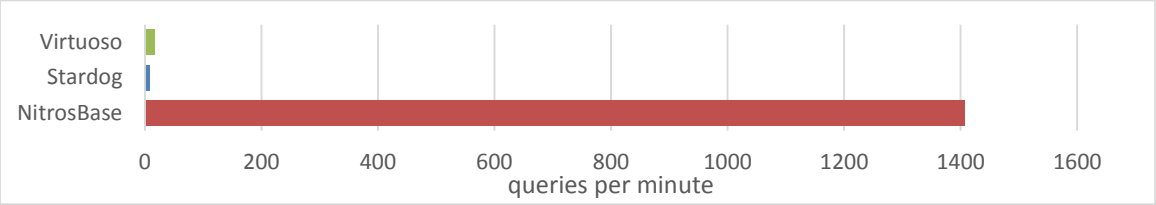
These 3 queries are almost identical to 20-22 queries. Additional condition added.

```
select ?x ?y ?title
where{
    ?x dcterms:issued ?y.
    ?x swrc:pages ?pages.
    ?x dc:title ?title.
    filter(?y >= "2005"^^xsd:integer && ?pages < "50"^^xsd:integer)
}
```

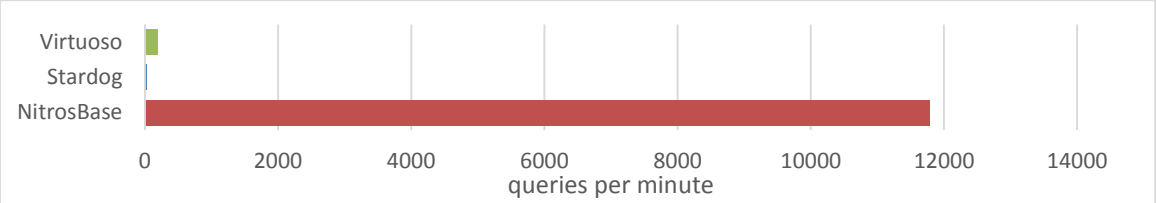
Query 30



Query 31



Query 32

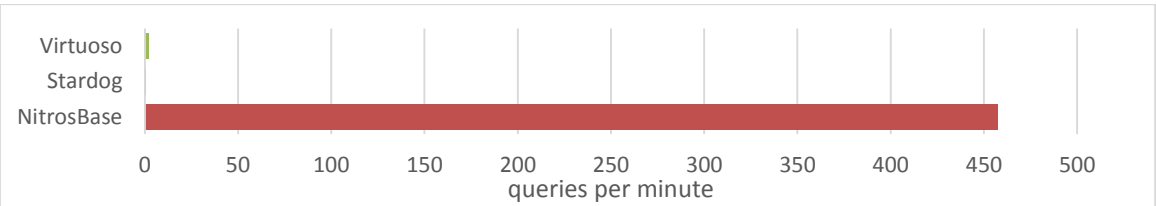


SIMPLE ORDER BY

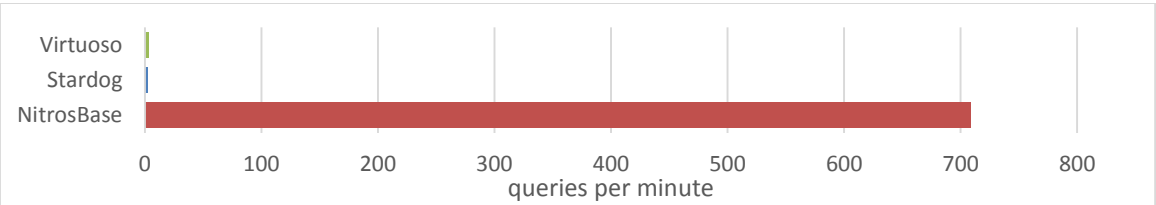
These 3 queries are almost identical to 20-22 queries. Result is ordered.

```
select ?x ?y ?title
where{
    ?x dcterms:issued ?y.
    ?x swrc:pages ?pages.
    ?x dc:title ?title.
    filter(?y >= "2005"^^xsd:integer)
}
order by ?title
```

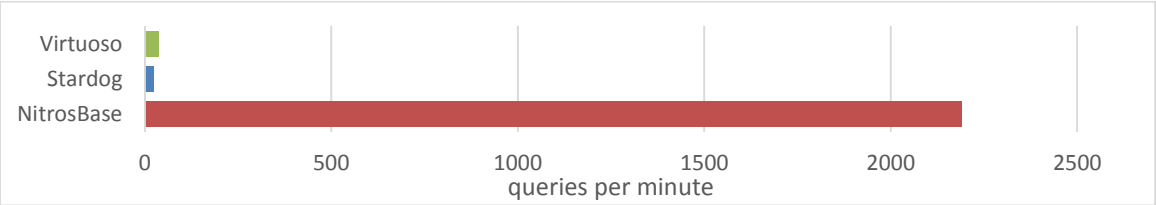
Query 40



Query 41

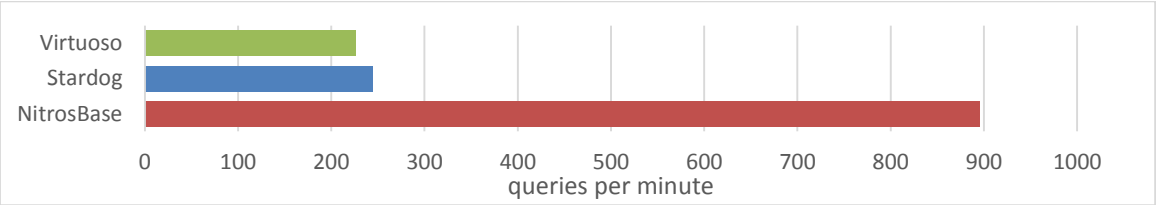


Query 42

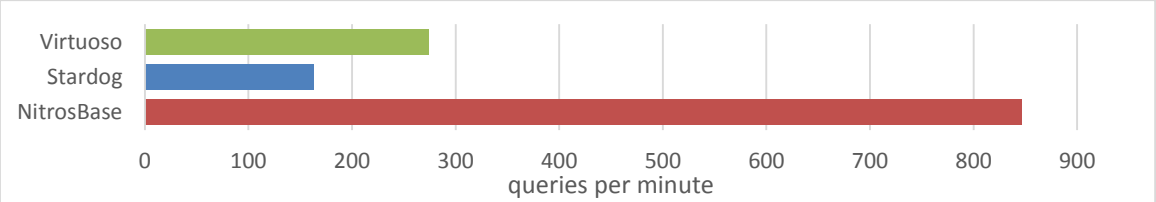


SIMPLE GROUP BY

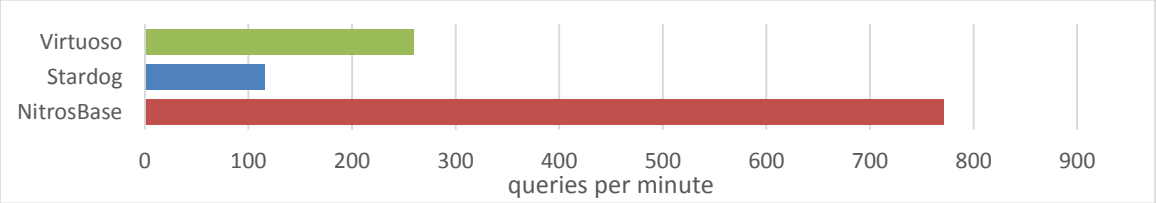
Query 50



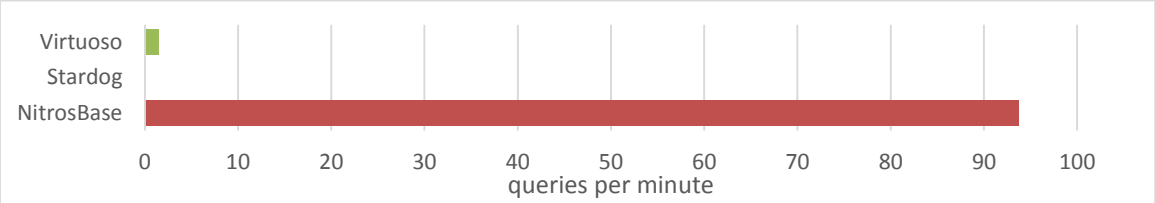
Query 51



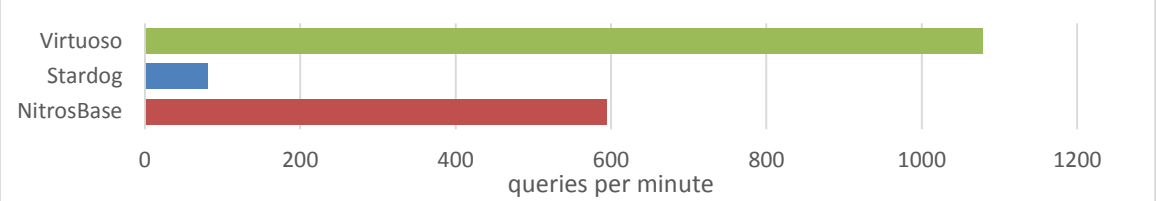
Query 52



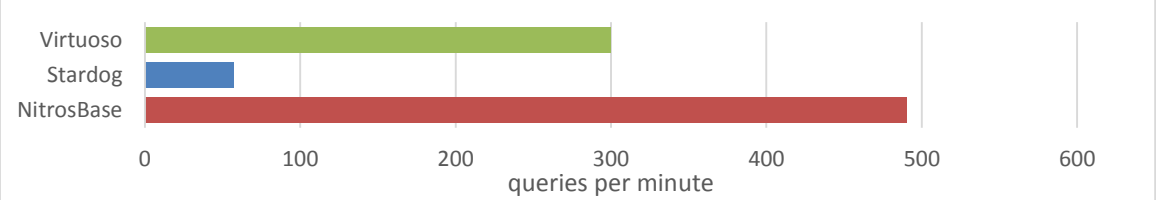
Query 53



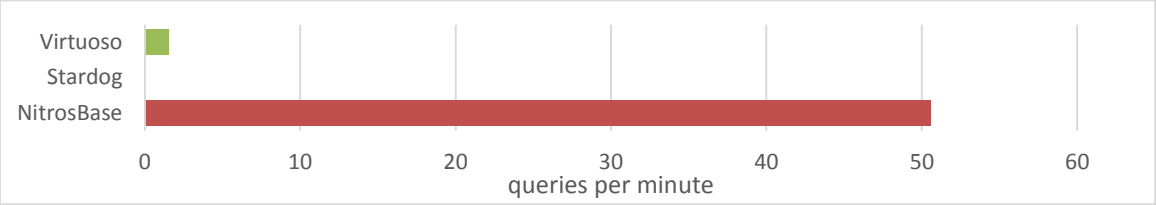
Query 60



Query 61

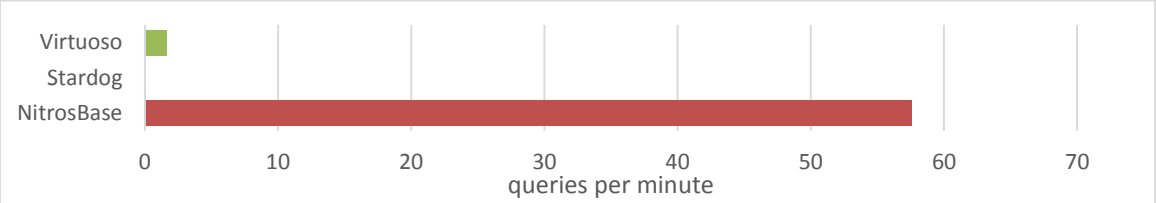


Query 62



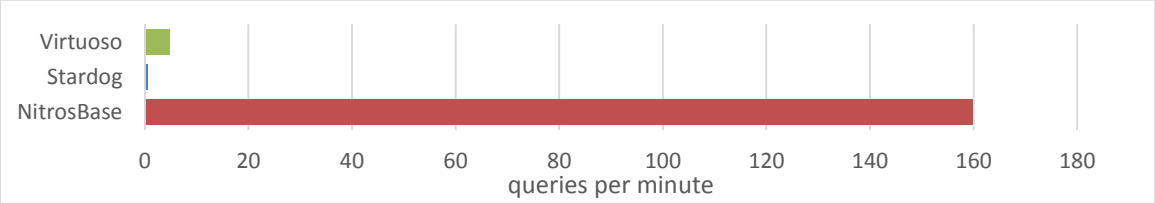
Query 63

Almost like 62 but we select distinct instead of group by



Query 64

Almost like 62 but simple filter added



Results discussion

Before we started these experiments, we had the belief that RDF databases are slow. The experiments revealed that RDF databases are developing and performance is growing. For example, query 20 and further results show that RDF storages perform fast and can compete with SQL databases. In future we plan joint testing of SQL and RDF databases.

Queries 60 and 61 revealed NitrosBase's optimizer problem. NitrosBase developers focused on complicated queries and overlooked the simple cases.

Comparison of the results showed that NitrosBase is performance leader on practically all queries. Virtuoso sure takes 2nd place. Stardog takes 3rd, but it does not mean that the last is poor. All 3 triplestores are indisputable performance leaders.

Note: When reviewing the results, it does not make sense to pay attention to 1.5-2 times result variation. This is natural fluctuation.

The result of our experiments is not official. This is just an attempt to "touch" three RDF tools and get a general idea of the factors affecting their performance.

Appendix 1

Experiment results are presented in the following tables. Each value presents time measured in milliseconds.

Computer 1:

Query	NitrosBase	Stardog	Virtuoso
q01	5	22997	518
q02	1583814	fail	140087354
q03a	54328	9725924	11418243
q03b	4765	86865	77513
q03c	2506	27813	3135
q05a	3125831	fail	21335902
q05b	545158	107931448	16236869
q06	1625924	fail	207752624
q07	66836	13866870	62765539
q08	1016	416458	48087
q09	1150789	15803823	3265959
q10	50	25663	13916
q11	8	2033861	1232251
q12a	22	8407547	1739
q12b	13	131085	26285
q12c	5	11252	265
q20	129008	170060861	21757596
q21	83476	23922854	13771229
q22	6894	2376668	1195975
q30	62915	11593611	5468326
q31	42680	7326679	3627272
q32	5091	1875197	306016
q40	131115	156046162	30620603
q41	84642	30246351	18751001
q42	27398	2637900	1580940
q50	67031	245866	265858
q51	70859	367479	219174
q52	77834	519474	230954
q53	640554	fail	41559420
q60	100910	750581	55667
q61	122351	1050396	200442
q62	1187102	fail	39410007
q63	1042391	fail	35891531
q64	375653	102587694	12743932

Computer 2:

Query	NitrosBase	Stardog	Virtuoso
q20	260229	fail	17 522 322
q21	161423	144349644	11388279
q22	13616	3312932	974756
q30	123017	14229527	5005780
q31	79049	9830377	3452072
q32	10817	2238886	278897
q40	267557	235022161	44628742
q41	156082	37805727	29586484
q42	48428	3515197	1480360
q50	134967	341642	255505
q51	125563	380910	388675
q52	148664	784128	414143
q53	925829	fail	31 149 544
q60	191870	930068	613562
q61	234778	1448012	490013
q62	1749368	fail	38 613 926
q63	1559766	fail	38 208 613
q64	554182	fail	12 797 403

Computer 3:

Query	NitrosBase	Stardog	Virtuoso
q20	125647	134912080	11135239
q21	80833	21170582	6763039
q22	6738	2132616	653323
q30	59480	9253996	2846559
q31	40430	6328763	1955788
q32	4496	1687837	227834
q40	127105	126812409	15445879
q41	82028	25280768	9257086
q42	24901	2338650	754312
q50	67181	229431	130079
q51	70772	269153	162227
q52	75824	385962	248089
q53	609846	fail	12242030
q60	98704	fail	235873
q61	123457	fail	284016
q62	1081411	fail	12745968
q63	925877	fail	12856960
q64	330951	fail	4309307

Appendix 2

The queries run in the experiments. Queries 1-12C are standard Sp2Bench queries and don't need to be listed here. You can download them from SP2Bench site (*http://dbis.informatik.uni-freiburg.de/content/projects/SP2B/docs/sp2b-v1_01-full.tar.gz*). Extended queries are below:

Prefix:

```
prefix dc: <http://purl.org/dc/elements/1.1/>
prefix dcterms: <http://purl.org/dc/terms/>
prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
prefix swrc: <http://swrc.ontoware.org/ontology#>
prefix foaf: <http://xmlns.com/foaf/0.1/>
prefix bench: <http://localhost/vocabulary/bench/>
prefix xsd: <http://www.w3.org/2001/XMLSchema#>
prefix person: <http://localhost/persons/>
```

q20

```
select ?x ?y ?title
where{
?x dcterms:issued ?y.
?x swrc:pages ?pages.
?x dc:title ?title.
filter(?y >= "2005"^^xsd:integer)
}
```

q21

```
select ?x ?y ?title
where{
?x dcterms:issued ?y.
?x swrc:pages ?pages.
?x dc:title ?title.
filter(?y >= "2010"^^xsd:integer)
}
```

q22

```
select ?x ?y ?title
where{
?x dcterms:issued ?y.
?x swrc:pages ?pages.
?x dc:title ?title.
filter(?y >= "2015"^^xsd:integer)
}
```

q30

```
select ?x ?y ?title
where{
?x dcterms:issued ?y.
?x swrc:pages ?pages.
?x dc:title ?title.
filter(?y >= "2005"^^xsd:integer && ?pages < "50"^^xsd:integer)
}
```

q31

```
select ?x ?y ?title
where{
?x dcterms:issued ?y.
?x swrc:pages ?pages.
?x dc:title ?title.
filter(?y >= "2010"^^xsd:integer && ?pages < "50"^^xsd:integer)
}
```

q32

```
select ?x ?y ?title
where{
?x dcterms:issued ?y.
?x swrc:pages ?pages.
?x dc:title ?title.
filter(?y >= "2015"^^xsd:integer && ?pages < "50"^^xsd:integer)
}
```

q40

```
select ?x ?y ?title
where{
?x dcterms:issued ?y.
?x swrc:pages ?pages.
?x dc:title ?title.
filter(?y >= "2005"^^xsd:integer)
}
order by ?title
```

q41

```
select ?x ?y ?title
where{
?x dcterms:issued ?y.
?x swrc:pages ?pages.
?x dc:title ?title.
filter(?y >= "2010"^^xsd:integer)
}
order by ?title
```

q42

```
select ?x ?y ?title
where{
?x dcterms:issued ?y.
?x swrc:pages ?pages.
?x dc:title ?title.
filter(?y >= "2015"^^xsd:integer)
}
order by ?title
```

q50

```
select (COUNT(*) as ?cnt) ?journal
where{
?x swrc:journal ?journal.
}
group by ?journal
```

q51

```
select (COUNT(*) as ?cnt) ?title
where{
?x swrc:journal ?journal.
?journal dc:title ?title.
}
group by ?title
```

q52

```
select (COUNT(*) as ?cnt) ?title ?issued
where{
?x swrc:journal ?journal.
?journal dc:title ?title.
?journal dcterms:issued ?issued.
}
group by ?title ?issued
```

q53

```
select (COUNT(*) as ?cnt) ?y ?title
where{
?x swrc:journal ?journal.
?journal dc:title ?title.
?journal dcterms:issued ?issued.
?x dc:creator ?y
}
group by ?y ?title
```

q60

```
select (COUNT(*) as ?cnt) ?issued
where{
?x dcterms:issued ?issued.
?x swrc:pages ?pages
}
group by ?issued
```

q61

```
select (COUNT(*) as ?cnt) ?issued
where{
?x dcterms:issued ?issued.
?x swrc:pages ?pages
}
group by ?issued ?pages
```

q62

```
select (COUNT(*) as ?cnt) ?y ?issued
where{
?x dcterms:issued ?issued.
?x swrc:pages ?pages.
?x dc:creator ?y
}
group by ?y ?issued
```

q63

```
select distinct ?y ?issued
where{
?x dcterms:issued ?issued.
?x swrc:pages ?pages.
?x dc:creator ?y
}
```

q64

```
select (COUNT(*) as ?cnt) ?y ?issued
where{
?x dcterms:issued ?issued.
?x swrc:pages ?pages.
?x dc:creator ?y
filter(?pages < "50"^^xsd:integer)
}
group by ?y ?issued
```