Configuring Virtuoso for Scale

Table of Contents

Background

OpenLink Virtuoso is an engine of many features. It incorporates a database engine, Web server, RDF quad-store, SPARQL processor, and the OpenLink Data Spaces (ODS) suite of applications for bookmarks, briefcase, wiki, webmail, etc.

Naturally, each of these features has its dependencies and consequences for resource usage, be that in-memory or on disk.

Here we present an overview of 3 possible configurations for Virtuoso.

For comparison, we use Virtuoso Open-Source Edition (VOS) compiled on Debian GNU/Linux ("Testing" distribution).

Structure of a Virtuoso Installation

A typical Virtuoso installation comprises the following files/areas:

Directory	Description	Files	Size
`/usr/bin/`	Vital operating binaries for server and command-line client	`virtuoso-t` `isql-v` `isql-vw`	9.2 M
`/usr/share/virtuoso/vad/`	ODS and other packages (rdf_mappers/sponger, tutorial, demo, isparql, etc.)	`rdf_mappers_dav.vad` `doc_dav.vad`	131 M
`/usr/lib/`	client drivers for ODBC & JDBC	`jdbc-2.0/virtjdbc*.jar` `virtodbc*.so`	12 M

`/usr/lib/virtuoso/hosting/`	Runtime loadable plugins for hosting and wiki markup	`wikiv.so` `mediawiki.so` `creolewiki.so`	3.6 M
`/var/lib/virtuoso/vsp/`	Splash-page for top-level web interface	`vsp/` `vsmx/` `images/` `frames/` `css/` `index.html` etc.	1.5 M
`/usr/share/doc/virtuoso-opensource/`	Misc README data for OS platform	`README.Debian` `README.CVS.gz` etc.	128 K
`/usr/lib/jena/`	Java class modules for using Virtuoso from Jena	`virt_jena.jar`	56 K
`/usr/lib/sesame/`	Java class modules for using Virtuoso from Sesame	`virt_sesame2.jar`	56 K
`/var/lib/virtuoso/db`	Default database directory	`virtuoso.ini` `virtuoso.db`	12 M

Operation

When the server is run against a given .ini file, it looks for a database at the name and location specified in that file. If none is found, the server will create an empty database with minimal system schema.

If it finds an appropriate directory containing *.vad files (set in the virtuoso.ini file during `make install'), Virtuoso will install the Conductor package by default.

From there on, the Virtuoso administrator is expected to use the Conductor, <http://localhost:8890/conductor/>, to install further packages such as the ODS application suite, etc.

After a checkpoint, a virtuoso.db file forms a portable unit encapsulating the entirety of a database instance, and can be moved between servers, so you can implement custom applications and schemas using the client interfaces (SQL/ODBC/JDBC) and avoid installing any of the packages, even the Conductor, should you so desire.

Configuration Options

The following parameters in a virtuoso.ini file defining a Virtuoso instance control the resource consumption and performance:

`[Database]`			
`DatabaseFile`	`virtuoso.db`	Filename of database file	docs

FileExtend	200	The amount of 8K-sized pages by which the database file automatically grows when the current file is not large enough.	docs
MaxCheckpointRemap	2000	Controls how many pages may be stored other than their logical page during checkpoints	docs
Striping	0	Enables the database file-striping mechanism	docs
[Parameters]			
ServerThreads	20	maximum number of threads (SQL+HTTP) used in the server - should be close to the number of concurrent connections	docs
CheckpointInterval	60	interval (minutes) at which Virtuoso will automatically make a database checkpoint	docs
NumberOfBuffers	2000	controls the amount of RAM (8K pages) used by Virtuoso to cache database files	docs
MaxDirtyBuffers	1200	The maximum number of modified buffers to store before writing	
MaxStaticCursorRows	5000	the maximum number of rows returned by a	docs

		static cursor	
FreeTextBatchSize	100000	the amount of text data processed in one batch of the free-text index during batch reindexing	docs
[HTTPServer]			
ServerThreads	10	Maximum concurrent HTTP sessions; must be less than the overall database ServerThreads	docs
MaxKeepAlives	10	A maximum number of HTTP sockets with KeepAlive connections	docs
KeepAliveTimeout	10	Timeout (s) before an idle HTTP connection is closed	docs
HTTPThreadSize	280000	Stack-size of an HTTP thread for handling connection and processing	docs
[Striping]			
Segment1	100M, db-seg1-1.db, db-seg1-2.db	Segment-specification for disk-striping	docs
Segment2	100M, db-seg2-1.db		
[SPARQL]			
ResultSetMaxRows	100000	Maximum number of rows in a SPARQL resultset	docs
[Plugins]			
LoadPath	/usr/lib/virtuoso/hosting	Directory in which to search for plugins	docs
Load1	plain, wikiv	Main ODS-Wiki markup parser	
Load2	plain, mediawiki	Auxilliary MediaWiki markup parser	

		module for ODS-Wiki	
`Load3`	`plain, creolewiki`	Auxilliary Creole wiki markup parser module for ODS-Wiki	
`Load4`	`plain, im`	ImageMagick plugin used by ODS-Gallery	
`Load5`	`plain, wbxml2`	WbXML plugin used by the SyncML package	
`Load6`	`plain, hslookup`	Required for some Sponger Cartridge operations	
`Load7`	`Hosting, hosting_php.so`	Module for hosting PHP scripts within Virtuoso	
`;Load8`	`Hosting,hosting_perl.so`	Module for hosting Perl scripts within Virtuoso (unsupported at present)	
`;Load9`	`Hosting,hosting_python.so`	Module for hosting Python scripts within Virtuoso (unsupported at present)	
`;Load10`	`Hosting,hosting_ruby.so`	Module for hosting Ruby scripts within Virtuoso (unsupported at present)	
`;Load11`	`msdtc,msdtc_sample`	For Microsoft XA transaction support	

Notes:

- Striping is an obvious way to control the amount of disk-space used; by default, striping is off (0) so the [Striping] section does not come into play.
- The checkpoint-interval setting is simply the amount of time for which a temporary database will grow before being checkpointed into the main virtuoso.db file, so a choice is

scenario-dependent; for a given incoming transaction rate, a short interval will give frequent smaller checkpoints while specifying a longer interval will make fewer, slower, checkpoints.
- The numbers of threads in the database engine as a whole and specifically allocated to the HTTP server will control performance, and each thread will cost a given amount of memory also.
- Naturally you can disable any or all unused plugins for further reduce the memory footprint; for example, if not running ODS-Wiki, you can remove the wikiv, mediawiki and creolewiki plugins.

We have a documentation page on tuning Virtuoso for RDF usage.

Analysis

Virtuoso provides the `status()` command, which may be executed through the SQL interface (e.g., isql-v(1)).

The resultset from this command is documented here, but we highlight specifically consideration of the `NumberOfBuffers` parameter; from `status()` output you will see how many buffers the server is actually using so you can tailor the allocated number accordingly.

Default

By default, the Debian virtuoso-opensource package enables all possible hosting options except Mono.

package .deb	58 M
server binary, /usr/bin/virtuoso-t	8.3 M
default database with Conductor installed	37 M
Virtual memory allocation	354 M
Resident memory used	125 M

Embedded / Minimal

The minimum that is required to run Virtuoso is the server executable (`virtuoso-t`, or in commercial edition, `virtuoso-iodbc-t`) compiled with as few options as possible, and the `virtuoso.ini` file. From there, the first time the server is run against the `virtuoso.ini`, it will create the empty database (`virtuoso.db`) with minimal schema.

The most important parameter to consider when optimizing for size is `NumberOfBuffers`.

By applying a few changes to `virtuoso.ini`, one can quite dramatically reduce the memory footprint:

```
[Database]
FileExtend          = 100                       ; down from 200
```

```
MaxCheckpointRemap = 1000                    ; down from 2000

[TempDatabase]
MaxCheckpointRemap = 1000                    ; down from 2000

[Parameters]
ServerThreads      = 5                       ; down from 10
CheckpointInterval = 10                      ; down from 60
NumberOfBuffers    = 100                     ; down from 2000
MaxDirtyBuffers    = 50                      ; down from 1200
SchedulerInterval  = 5                       ; down from 10
FreeTextBatchSize  = 1000                    ; down from 100000

[HTTPServer]
ServerThreads      = 2                       ; down from 5
KeepAliveTimeout   = 5                       ; down from 10
HTTPThreadSize     = 10000                   ; down from 280000

[Client]
SQL_PREFETCH_ROWS  = 10                      ; down from 100
SQL_PREFETCH_BYTES = 4096                    ; down from 16000

[Replication]
ServerEnable       = 0                       ; changed from 1

[Zero Config]
;ServerName          = virtuoso (SAUCE)     ; commented-out

[SPARQL]
ResultSetMaxRows   = 10000                   ; down from 100000

[Plugins]
; ...                                           ; comment-out all
plugins
```

Result:

default empty database with no Conductor installed	12M
Virtual memory allocation	150M
Resident memory used	70M

Executing `status()` shows that 1000 buffers are allocated and only 270 are in use.

Comparison with LAMP

On the same machine, we installed Apache 2.x and MySQL 5.0 server using standard Debian GNU/Linux packages:

```
bash$ sudo apt-get install mysql-server apache2
```

This is not quite comparing like with like; Virtuoso includes not only HTTP and SQLinterfaces, but also a complete RDF Quad-store, SPARQL processor, free-text indexer, etc. However, out-of-the-box MySQL and Apache consumptions compare to the above-tuned Virtuoso as follows:

	Apache	MySQL	Virtuoso
package *.deb	45.6 M	-	58 M
server binary	0.3 M	7 M	8.3 M
default empty database	-	21 M	12 M
virtual memory allocation	229 M	123 M	350 M
resident memory usage	2.8 M	15 M	70 M

Enterprise-wide

In larger installations, the NumberOfBuffers parameter should be increased, but there is no point in making the process so large it has to swap. Therefore we recommend about 60% memory should be allocated to buffers.

As an example, we consider two instances: our own web-server, www.openlinksw.com (running on Debian GNU/Linux), and DBpedia.org (running on Sun Solaris).

	www	DBpedia
Virtual memory allocated	2734M	6257M
Resident memory	662M (65%)	6216M (38%)
Striping	0	0
CheckpointInterval	120	60
NumberOfBuffers	20000, 18511 used	100000, 99964 used
MaxDirtyBuffers	8000	40000
MaxCheckpointRemap	16000	80000
FreeTextBatchSize	100000	100000
ServerThreads	10	1000
HTTP ServerThreads	250	100
HTTPThreadSize	280000	10000000
Plugins	only wikiv	wikiv, imagemagick, wbxml2
SPARQL ResultSetMaxRows	1000	1000

References

- Virtuoso performance diagnostics
- Virtuoso performance tuning