Tuning the server

In This Chapter

Java Virtual Machine (JVM) Tuning

Java version
Servoy supports both Java 5 and Java 6 (for the Application Server, the Smart Client and Servoy Developer). As Java 6 has numerous performance improvements, it is preferred over Java 5.

If the hardware in which Servoy is running is 64bit, make sure to also run a 64 bit Java Virtual Machine. When in doubt which version is used, check the Servoy Admin page, under "Servoy Server Home" > "Servoy Server Status" > "JVM Information":

- 64 bit: java.vm.name=Java HotSpot(TM) 64-Bit Server VM
- 32 bit: java.vm.name=Java HotSpot(TM) Client VM

An 32 bit JVM will allow a max memory of 2 Gb in total, In order to run with larger heap sizes (over 2 Gb) you have to use 64 bit JVM on a 64 bit OS

Java Virtual Machine Server mode
Java has 2 distinct operating modes: client mode and server mode. When running in server mode, the startup of the Java process will take longer and initial execution of code will be slower, but after a while the application will perform faster. The server mode is best suited for the Servoy Application Server.

Whether or not a Java Virtual Machine supports client and/or server mode depends on the hardware, see http://download.oracle.com/javase/6/docs/technotes/guides/vm/server-class.html.

If the JVM supports servermode, it will automatically detect if the machine it is running on is a "capable" machine and automatically run in server or client mode depending. The definition of "capable" may differ from Java version to Java version. For Java 5/6 this means a 2CPU, 2Gb memory machine.

Checking whether or not a Servoy Application Server runs in client or server mode can be seen on the Servoy Admin page, under "Servoy Server Home" > "Servoy Server Status" > "JVM Information":

- Server mode: java.vm.name=Java HotSpot(TM) 64-Bit **Server VM**
- Client mode: java.vm.name=Java HotSpot(TM) **Client VM**

It is possible to enforce the Servoy Application Server to run in server mode (assuming the JVM is capable to operate in server mode), an extra startup argument can be added to the startup of the Servoy Application Server.

In servoy_server.sh/bat:

```
java -server -Djava.awt.headless=true .....
```

When using the Service wrapper:

```
# Java Additional Parameters
wrapper.java.additional.1=-Djava.awt.headless=true
wrapper.java.additional.2=-Duser.dir="C:\Servoy\application_server"
wrapper.java.additional.3=-Djava.io.tmpdir="C:Servoy\application_server\server\work"
wrapper.java.additional.4=-XX:MaxPermSize=128m
wrapper.java.additional.5=-server
```


On Windows 32 bit environments, the Java Runtime Environment does not include support for server mode. In order to take advantage of server mode optimizations on 32 bit Windows systems, it is required to install Java Development Kit (JDK)

Memory

See Memory Management

Database Connection Pool Tuning

See Connection Pooling in the Database Connections chapter

Database Tuning

Architecture

In order to get the best possible performance, the communication between the Servoy Application Server and the database needs to be as fast as possible. The fastest option would be installing the database engine and the Servoy Application Server on the same machine, whether or not this is possible depends on the dimensions of the physical server and the resource requirements of both the database engine and the Servoy Application Server.

When opting for dedicated servers for both the database engine and the Servoy Application Server, the connection between them should be as fast as possible, preferably a direct 1Gb cable connection between the two machines.

Database tuning

When experiencing performance issues the "Performance Data" page of the Servoy Admin page can be analysed to find slow performing SQL statements, or SLQ statements that are fired often.

The page provides an overview of all queries fired at the database. The performance data is sorted by the total time taken to perform the query, being the result of the average time * number of executions.

At any given time regular SQL statements (Type = Relation, Load foundset) should have an average execution time below 25 milliseconds.

Especially for SQL statements that are fired often and take up significant time it should be checked if it's possible to optimize the database, for example by creating the relevant indexes.

Foundset loading Tuning

Servoy loads a foundset by first retrieving the PKs from the database. This retrieval is done in chunks: when the foundset is initialized, the first chunk is retrieved, consecutive chunks are retrieved when needed.

The actual record data is also retrieved from the database in chunks.

The size of the different chunks are controlled by 3 properties that can be set in the servoy.properties:

Setting	Default Value	Description
servoy.foundset.pkChunkSize	200	Chunk size for foundset PK retrieval
servoy.foundset.chunkSize	30	Chunk size for record data retrieval
servoy.foundset. initialRelatedChunkSize	2 * servoy.foundset. chunkSize	Chunk size for related record retrieval For the initial load of related records both the PK's and data are retrieved in one query

The values can be modified in an attempt to increase performance. The optimal values differ per application, thus no guidelines are available for alternative values. Tuning needs to be done by altering the values and monitoring the performance afterwards. The defaults are set based on averages.

Web Client Tuning

Content compression

In order to minimize the size of the content that the browser has to download for the Web Client, the application server serving the content can compress the content. The Servoy Application Server is configured to automatically compress content.

It is also possible to handle the compression outside of the Servoy Application Server, for example in an Apache HTTPD Server placed between the Servoy Application Server and the outside world. The advantage of this is that Tomcat's compression options are limited, so an external HTTP server offers more options.

When it comes to Apache HTTPD server, there are two options:

1. Compression using mod_deflate
2. Compression using gzip

When it comes to compression, there are two factors that come into play, the time it takes to do the actual compression and the rate of compression achieved, resulting in quicker downloads to the browser due to smaller size. This is primary difference between mod_deflate and gzip in Apache HTTPD: while gzip gets a better compression rate than mod_deflate, it takes more time and CPU to perform the compression. In the case of Servoy Web Clients, that, due to it's Ajax driven design, performs a lot (albeit small) requests to the server, the overall performance will be better using mod_deflate.

Details on setting up mod_deflate on an Apache HTTPD server can be found here: http://httpd.apache.org/docs/2.0/mod/mod_deflate.html

When compression is handled by outside the Servoy Application Server, it is advised to turn of compression inside Servoy. This can be doen by modifying the "server.xml" file located in {servoy_install}application_server\server\conf\. and locate the Connector:

```
<Connector port="8080"
protocol="HTTP/1.1"
maxThreads="500" connectionTimeout="60000"
redirectPort="8443" useBodyEncodingForURI="true"
compressableMimeType="text/html,text/xml,text/css,text/javascript,text/plain" compression="on"
compressionMinSize="2048"/>
```

and removing the following atttributes:

- compressableMimeType
- compression
- compressionMinSize

Save the edits and restart the Servoy Application Server to let the configuration changes take effect.

Servoy Application Server log

At all times try to keep the Server log free from javascript application errors/excpetions, all application exceptions point to possible problems inside the application.

- Out of memory errors means the heap size is configured to small.
- Nullpointer errors indicate Servoy codebase problems, incase these are noticed file a support case containing the log entry.
- Network error are only listed as non critical (orange) which may point to (possible) networks (config) problems, when many errors are continuous listed these should definitely be investigated.