Debugging Load and Memory Leak Issues

Debugging problems during production is very important as there are many potential contributing factors. Users may need to isolate a few parameters to identify the cause.

Possible causes:

a. Third party code that is deployed with the server like JDBC Drivers, codes like Javascript
b. Java VM version. See if patch is present.
c. Load of the report rendering.
d. Insufficient memory configured i.e. the total Java RAM configured exceeded the Server production RAM. This could lead to production slow down.
e. Improper sizing
f. Our code defects etc

The cause of the issue could be a combination of these factors.

1. Environment information

For us to help customers do further investigation, customers should send the following information to us (support@elixirtech.com)

Elixir Repertoire Server
- Version
- Running method(startServer.bat? startServer.sh? System Service used?)
- Configuration such as bat/sh file (Heap memory setting), wrapper.conf for System Service configuration
- Type of rendering when the incident occurs (PDF? Excel? etc).
- Log files when the incident occurs (server.log, activity.log, etc).
- The template files (and corresponding data source if any)

Other information
- Java VM version (1.5? 1.6?)
- Database Server, Drivers

System architecture
- CPU
- Operating system 32/64bit
- Number of servers/ load balancer and the configuration.

Load during the incident:
- number of concurrent clients
- number of pages per report renders etc.

2. Log information

- Identify load at the time of rendering. Errors can be obtained from logs.
- Check which report is being rendered to see if the issue is replicated with the same report.

Resolution: You may need to increase the JVM memory etc.

3. Review the current System/ Network architecture for potential bottlenecks

- Check if there is bottleneck in the current system or network design
- Check if there is sufficient memory allocated to the system. Typically you should keep 20% - 30% of the memory for Operating System functions.
- Tuning maybe needed
http://java.sun.com/performance/reference/whitepapers/tuning.html

4. Get heap and thread memory dump

http://java.sun.com/javase/6/webnotes/trouble/TSG-VM/html/memleaks.html

The command (can be found in the above link):

-XX:+HeapDumpOnOutOfMemoryError

You can collect the heap dump and send it to the development team for analysis.

- Migration issue.
The older ER4 version, uses a multi-process architecture while the newer ER7 has a single process architecture with many rendering threads. Due to this architectural difference, you may need to resize the servers.

Other useful tools for monitoring memory:

http://java.sun.com/developer/technicalArticles/J2SE/monitoring/

5. Some Suggested Resolutions

1. Check if the real cause is due to server misconfiguration.

2. Tune the server and apply Java best practices.
http://java.sun.com/performance/reference/whitepapers/tuning.html

3. Modify architecture. (after you have done point 1)

4. Tune the report and the various configurable entities such as cubes.
- For the report, you can turn off page count to keep the memory usage low if you have no need to retain page count.
- Turn off Table mode with SQL pre-sorted to match the grouping order in the template. Set grouping to None for sorting, if you do not need sorting and grouping.
This should cut down the memory requirement down to the theoretical minimum, i.e just 1 record at a time, as there is no need to cache data required for sorting, in memory.

5. A temporary solution is to restart the server so that the memory issue is cleared while we try to debug the problem.

6. Mapping version 4 memory to V7 for reporting

A. Version 4 consists of

i) One listener process and memory is set in the batch file.
ii) N child processes and memory per child process is set in the configuration file.

Total memory needed is 1 x P (listener process) + N x P (child processes)

Though we need X amount of memory for the whole report server, the individual Java process will never exceed the memory value set for that process. If the largest child process is set at 256MB, the individual child process will never grow beyond 256 MB even at full load. Instead, the memory management is left to the OS to handle natively. Hence, you do not have to worry about the Java tuning parameters and Garage Collection (GC).

It is easier to estimate the memory needed per report as only one report is rendered in the process.

Hence, typically for a ER4 dual engine server

A) 1 Listener process == 128 MB
B) 2 Engine == 256 X2 MB

The total memory is 128 + 256X2 = 768MB. But it is still "safe" if you have only 512MB on your hardware server as a single Java process does not exceed 256MB.

B) Version 7 consists of only threads to handle the incoming request.

i) N rendering threads for report (only). There is also a thread for queue you can ignore it.

Total memory needed is N x P (Reporting threads)

For threaded application, at full load if the server is set for 512MB (per thread 512MB), then all of 512MB needs to be committed for rendering the reports concurrently. If the hardware server has only 512MB it will run out of physical memory to allocate. The Garbage Collector will kick in and compete with the OS to free memory.

The only solution in this case is to fine tune the server Java VM Garage collection options. We need to lower the memory to 80% of the total, as some of the memory is shared between the threads, when rendering processes. Do this is only after load testing.

If except for Version 7, there are other services running such as scheduling etc, it will also affect the server memory requirement. If it is just a plain report it can be disabled. The complication comes in when running Ad Hoc Cube or Perceptive. There are more things to consider.

Example

To translate Version 4 Server to 2 Threaded engine V7 server which is

A) Dual ER4 server has 1 Listener (128MB) + 2 child processes (256MB per process) which work out to 768MB

to

B) Version 7 which has *2 Threads ==256 X2 MB* == 512MB

This mean effectively for 1 Java process, you will need 512 MB. You will have to allocate the actual physical memory for it. Otherwise you will hit into resource contention problem. We can only hope there will be a mixture of small and large report rendering. However, it is possible to have a long rendering report which can grab all the resources. This will be an issue when a very large report is being rendered...512MB per report rendering.

For version 4, it is safe to have a 2GB server with 5 child processes of 512MB each. However, for version 7, this means that you will allocate 5 x 512MB of memory i.e. 2560MB is needed. This can cause two problems:

a. It exceeded hardware memory
b. If the OS is 32bit and has only 2GB ram then you cannot increase your hardware RAM beyond that limit.

The only way is to reduce the concurrent threads so that the total memory is at 70% below that of hardware memory i.e. 3 reporting threads instead of 5.