

Solution for JQM crash caused by pdls on a loaded server

This technical update includes information related to crash of JQM server during heavy load and submissions of `pdls` command. The purpose of this technical update is to provide information about the root cause of the problem and workaround to fix the problem using environment settings.

Platforms Affected

All

Problem

During heavy load the JQM may go into unknown status or generates a core dump if the user tries to execute `pdls` command on a bunch of jobs.

This problem happens when JQM processes try to allocate memory more than the specified data segment size, leading to process running out of memory and JQM crash. Following are a set of options which you can try to avoid the problem.

Increase the data segment stack size

You can increase the data segment stack size by using `ulimit` command. Each operating system uses a separate data segment stack size. The following command will help in finding the current data segment stack size

```
ulimit -a
```

Example output for `ulimit -a` command on Solaris:

```
$ ulimit -a  
time (seconds) - unlimited  
file (blocks) - unlimited  
data (kbytes) - 96000  
stack (kbytes) - 8192  
coredump (blocks) - unlimited  
nofiles (descriptors) - 256  
vmemory (kbytes) - unlimited
```

In the above example, the data segment size is 96MB. So, if the `pdls` operation on a heavily loaded JQM is executed and if the JQM process tries to allocate memory more than the specified data segment stack size, the server process will go to unknown state leading to failure of jobs and may dump core

Please check the help information of `ulimit` command information on each operating system to change the data segment stack size.

Use indexed attributes while making pdls request

The JQM process buffers all data for the request and sends to the output when you make a `pdls` request on JQM. It is suggested to avoid making `pdls` request which could potentially return huge results.

For example: `pdls -c job -a all jqm:`

or

```
pdls -c job -a all -x"-job-status retained" jqm:
```

HP-UX specific Case

HP-UX uses different ARENA's for allocating memory for each thread. Due to this reason, for every `pdls` request there could potentially be an increase in the JQM memory usage if the request is served by different thread each time. However, the increase would be limited to the number of threads used by the JQM server process (Ex: 10 worker threads), after which JQM memory should not increase or at least should not be as much as it did for previous requests. This could be the reason for the crash only after 10 `pdls` and not during the first one itself for the same number of jobs in it's database.

This behavior of HP-UX memory allocator can be altered using `_M_ARENA_OPTS` environment variable.

Example:

```
export _M_ARENA_OPTS=1:8
```

Restart the JQM server and this would cause JQM server process to use single ARENA for all memory allocation requests preventing JQM server process memory from growing for any subsequent requests.

NOTE: The same can be achieved for a specific server by setting this variable in `-server-environment-variables` attribute for a server. Setting of `_M_ARENA_OPTS` variable may adversely affect the performance of an application, as per HP-UX manual for “malloc”, use of different ARENA for each thread leads to better performance.

The `M_ARENA_OPTS` can also be set for a specific server (JQM or any other) by using `-server-environment-variables` attribute and can be set using the following command

```
config_server -u -x"-server-environment-variables  
\"_M_ARENA_OPTS=1:8\"" jqm
```

Conclusion

With all this being said, the JQM design has a limitation where it buffers all the output for pdls in it’s memory prior to marshalling the data to the pdls client. This could cause JQM to crash for large number of jobs. The only way to avoid the crash is to increase “data segment size” (this can only reduce the probability of server crash) or avoid issuing pdls on bunch of jobs.

