Recommendations for running large job submissions using GT4 and Condor-G

The following observations had been made with jobs with the following characteristics:

- 2 MB stageIn data
- 2 MB stageOut data
- Action of the job: sleep a duration between 3-6 seconds and then copy 2 files (of size 2MB)

1 Server-Side

1. Start the GT4 container with 1024 MB memory to avoid OutOfMemory exceptions, especially if you use burst submission because then many jobs will be managed by the container concurrently.
2. Clean up all tables of the RFT database if you have an RFT database that was used by RFT from a GT version older than 4.0.3. All entries can be removed. The tables (especially the `restart`-table) may contain a huge number of records, depending on the frequency of use of RFT, because in older versions of RFT these records had not been removed after processing transfers.
   This can reduce job throughput considerably.
   SQL command to remove all records of the `restart`-table after a successful login to the database system and selection of the database `rft_database`:

   ```
   delete from restart;
   ```

3. For GT versions older than 4.0.3:
   Update RFT from globus_4_0_branch to have
   - efficient usage of the database connection pool
   - efficient table handling in the RFT database
   - no over-synchronization
4. Make sure that the directory `.globus` in the home directory of the user who runs the GT4 container, where the persistency data is stored, is not located on a mounted partition, but on the local disk. Persistence data located on an NFS-mounted partition for example can have big (negative) impact on performance.

2 Client-Side

If jobs are submitted using Condor-G:

1. Add the following line in the first `log4j.properties` file in the Java classpath:

   ```
   log4j.appender.A1.Target=System.err
   ```

   after

```
log4j.appender.A1=org.apache.log4j.ConsoleAppender
```

In my case it was `$GLOBUS_LOCATION/log4j.properties`. If there's no GT4 installation create a `log4j.properties` file in the first location of your Java classpath.
The gridmanager from Condor uses the GT class `GramJob` and will probably fail and restart every now and then if `GramJob` logs to `stdout` instead of `stderr` and this causes communication errors between Condor and the GT.

2. Update `$CONDOR_INSTALL/lib/gt4/lib/axis.jar` with a new version of axis.jar from globus_4_0_branch to avoid thread safety problems which caused

   - Jobs to keep stuck every now and then in concurrent job submissions, especially in a multi processor environment
   - Generation of duplicate UUIDs of jobs in the GT container. Duplicate UUIDs cause jobs to fail.

   I think Condor didn't upgrade to the new axis.jar so far (16/02/2007).
3. Create credentials with the default duration (12 hours).
   In case of long running tests Condor refreshes these credentials automatically, but not if the duration is a non-default one (and this causes errors in the communication between GT and Condor).
   (Although I still experienced problems with long running tests)
4. Make the following settings in `$CONDOR_CONFIG` to have good job throughput:

   - `GRIDMANAGER_MAX_SUBMITTED_JOBS_PER_RESOURCE = 40`
     `GRIDMANAGER_MAX_PENDING_SUBMITS_PER_RESOURCE = 40`
     These settings lead to throttled submission by Condor-G and avoid burdening the GT container and leads to better job throughput behaviour.
   - `GRIDMANAGER_JOB_PROBE_INTERVAL = 3600`
     This causes Condor-G to poll for job states only once an hour and to mostly use the notification mechanism. This improves performance. Too much polling leads to worse job throughput behaviour in the container especially if burst submission is used because then many jobs will be managed by the container concurrently.