Kepler - Bug #5429

improve default provenance store performance

06/24/2011 01:13 PM - Derik Barseghian

Status:	New	**Start date:**	06/24/2011
Priority:	Normal	**Due date:**	
Assignee:	Daniel Crawl	**% Done:**	0%
Category:	provenance	**Estimated time:**	0.00 hour
Target version:	reporting-2.X.Y	**Spent time:**	0.00 hour
Bugzilla-Id:	5429		

Description

Currently there can be some big performance penalties when using kepler with provenance turned on (by default using hsql). It would be great to improve these.

Unless noted, references to workflow execution times below refer to the reap GDD wf set to process 200days of data:
https://code.ecoinformatics.org/code/reap/trunk/usecases/terrestrial/workflows/derivedMETProducts/growingDegreeDays.kar

I see/saw a few issues:

-1) at one point I mentioned kepler shutdown was taking a very long time. This isn't an issue anymore, shutdown seems near instant.

0) the pre-initialize stage of workflow execution can take a very long time and grows longer w/ each subsequent execution when running with a provenance store that's large. E.g. up to 15m.
Dan's fixed this issue, I believe w/ r27746. Pre-init is now close to instant or just a few seconds.

1) execution of the workflow w/ provenance off takes a few seconds. With provenance on, it takes about 4min to run the first time with an empty provenance store.

2) subsequent executions of the same workflow take longer to run.
E.g. Here are the execution times of 9 runs of the workflow on 2 different machines:
10.6 macbook 2.2ghz intel core 2 duo w/ 4gb RAM:
4:01, 4:03, 3:57, 7:43, 8:07, 8:01, 8:33, 8:10, 8:33,
ubuntu 10.04 dual 3ghz w/ 2gb RAM:
4:03, 4:13, 4:32, 9:13, 12:32, 8:08, 9:54, 9:06, 11:53

3) startup time can take a very long time when the prior Kepler invocation ran data/token intensive workflows. I believe what's happening is hsql is incorporating the changes in the log file into the .data file. I think something's happening w/ the .backup file too. The data file slowly grows very large (a lot more than by 200mb), and finally the log file drops to near 0, and then the data file decreases in size to a size larger than where it started. I think with the default log file max size of 200mb, startup can take on the order of 10-20m. I've tested w/ a variety of log file sizes. Making it dramatically smaller, e.g. 5mb, dramatically improves startup time, but comes at a huge workflow execution time penalty (~20m to run the wf), so this is an unacceptable fix. The execution penalty starts happening when the log file max size is set smaller than about 100mb. With a 100mb log file, startup is still very slow.

One thing I've found that improves execution time performance is increasing the 'memory cache exponent' setting (hsqldb.cache_scale) from the default of 14 to the max of 18. This setting "Indicates the maximum number of rows of cached tables that are held in memory, calculated as 3 **(2***value) (three multiplied by (two to the power value)). The default results in up to 3*16384 rows from all cached tables being held in memory at any time."
With a 200mb log file max size, and cache_scale=18, the first run of the workflow takes about 2:17.

History

#1 - 06/24/2011 06:05 PM - Derik Barseghian

We're not explicitly doing a SHUTDOWN of the provenance DB when we quit Kepler, and "the log is never reused unless there is an abnormal termination, i.e. the database process is terminated without SHUTDOWN, or it was terminated using SHUTDOWN IMMEDIATELY." Doing a SHUTDOWN COMPACT of a large provenance store (~500mb) takes a long time, but then startup is near instant. A plain SHUTDOWN is significantly faster, and startup seemed to be just as fast. I didn't time how long SHUTDOWN took, but it seemed a lot faster than our current startup time, so I believe this would be an improvement. Dan's going to try to add a SHUTDOWN call to kepler on quit.

Two tangential notes:
1) At the end of the provenanceDB.script file is:
SET WRITE_DELAY 10
even though we're now simply setting WRITE_DELAY true, which should use the default of 20s.

2) We have cache_file_scale=1. The hsql 1.8 guide makes it sound like this limits the data file to 1gb, and yet I've had provenance stores bigger. I'm trying to see if hsql will truncate my DB...in which case we should probably raise this to the max of 8gb, I guess.

#2 - 06/28/2011 03:16 PM - Derik Barseghian

After a lot more testing with the cache_file_scale and hsqldb.cache_scale options, and for the GDD workflow processing 200days, the ideal setting seems to be about:

hsqldb.cache_size_scale=8 (the default. 8-10 is fine, below or above this range is worse)
hsqldb.cache_scale=18 (max, 14 is default)

Changing from running the provenance database in server mode to file mode also provides a significant execution time performance boost - the first run of the workflow takes a little less than 1min.

With cache_file_scale=1, my provenance store can grow larger than 1gb. I guess this setting means 2gb. I've yet to get a store that large to see what happens...

#3 - 07/06/2011 05:25 PM - Derik Barseghian

Did more research yesterday and found some changes to hsqldb post 1.8 that sound valuable wrt our shutdown performance issue w/ our large database where the entire

Found this change comment somewhere:

New features for incremental backup of the .data file allow speedy handling of huge databases up to 16GB.

and this:
----
22 Feb 2011 - SVN 4096 - 2.1.0 rc4
-- changes to avoid creation of new .data files at checkpoint and allow incremental enlargement of this file

and 2.0 has this new setting:

SET FILES BACKUP INCREMENT

set files backup increment statement

<set database backup increment statement> ::= SET FILES BACKUP INCREMENT { TRUE | FALSE }

Older versions of HSQLDB perform a backup of the .data file before its contents are modified and the whole .data file is saved in a compressed form when a CHECKPOINT or SHUTDOWN is performed. This takes a long time when the size of the database exceeds 100 MB or so (on an average 2010 computer, you can expect a backup speed of 20MB / s or more).

The alternative is backup in increments, just before some part of the .data file is modified. In this this mode, no backup is performed at CHECKPIONT or SHUTDOWN. This mode is prefered for large databases which are opened and closed frequently.

The default mode is TRUE. If the old method of backup is prefered, the mode can be set FALSE.

Only a user with the DBA role can execute this statement.

http://hsqldb.sourceforge.net/doc/2.0/guide/deployment-chapt.html

#4 - 07/07/2011 04:22 PM - Derik Barseghian

On the issue of max data file size (hsqldb.cache_file_scale=1), we are currently limited to 2gb. I've reached this with one of my stores. The likely scenario is mid-execution you start getting exceptions like:
java.io.IOException: S100 Data file size limit is reached in statement...

The run in the WRM will continue to show "running..." because the error can't get recorded.
Subsequent run-row deletes take an extremely long time and don't seem to work, there's a lot of activity with the log file, but they don't disappear until you restart kepler (which also takes an extremely long time).

To change max data file size, certain conditions should be in place, see:
http://hsqldb.org/doc/guide/ch04.html

Changing the size to 4gb (max for fat32) probably isn't even desirable until we upgrade hsql and can leverage the SET FILES BACKUP INCREMENT feature.

#5 - 11/30/2011 12:59 PM - Derik Barseghian

Iirc to make significant further gains in performance we need to move to HSQL 2.x, which Dan and I both made unsuccessful attempts at, and which seemed like will require a fair bit of work, including provenance schema changes. Retargeting, and reassigning to Dan.

#6 - 03/27/2013 02:30 PM - Redmine Admin

Original Bugzilla ID was 5429