Infrastructure - Task #7945

Bug # 7933 (New): d1-index-task-processor stops processing task queue

D1-index-processor should be shut down soon after stop given

2016-11-29 19:26 - Jing Tao

Status:	Closed	**Start date:**	2016-11-29
Priority:	Normal	**Due date:**	
Assignee:	Rob Nahf	**% Done:**	100%
Category:	d1_indexer	**Estimated time:**	0.00 hour
Target version:	CCI-2.3.1		
Milestone:	None	**Story Points:**	
Product Version:	*		

Description

Currently, the d1-index-processor will be shut down until current index-tasks queue be completed. It may take a long time. We should put those index tasks which haven't done back to the d1-index task table and finish the current running task , then shut down it.

Associated revisions

Revision 18528 - 2017-01-13 01:32 - Rob Nahf

refs #7945: Added logic to cascade interrupts to the ExecutorService when stop command is given. Needs testing. Needs better task cleanup logic.

Revision 18528 - 2017-01-13 01:32 - Rob Nahf

refs #7945: Added logic to cascade interrupts to the ExecutorService when stop command is given. Needs testing. Needs better task cleanup logic.

Revision 18544 - 2017-01-18 22:37 - Rob Nahf

refs #7945: Fixed logic that shutsdown the ExecutorService and returns the canceled tasks to NEW.

Revision 18544 - 2017-01-18 22:37 - Rob Nahf

refs #7945: Fixed logic that shutsdown the ExecutorService and returns the canceled tasks to NEW.

Revision 18546 - 2017-01-19 01:04 - Rob Nahf

refs #7945: minor code cleanup.

Revision 18546 - 2017-01-19 01:04 - Rob Nahf

refs #7945: minor code cleanup.

History

#1 - 2016-12-02 00:25 - Dave Vieglais

Need to diagnose what's going on here. If it involves too much refactoring then we should move to be part of the 2.4 release.

Immediate task is to identify and document why index processor takes many hours to shut down.

#2 - 2016-12-14 18:16 - Rob Nahf

- Category set to d1_indexer

The BATCH_UPDATE_SIZE constant found in class org.dataone.cn.index.processor.IndexTaskProcessor, doesn't seem to be used anywhere - only in commented out code. It looks like the ProcessorJob gets all of the index tasks, and interrupts are handled such that new jobs are not scheduled, but existing ones finish up.

I'm not certain, but it looks like the task is only removed from the queue when the item is fully processed. This means that the index processor is very interruptable, with the only consequence being that current in-memory state is lost - maybe including notes on whether a resource map is ready to be

processed.

#3 - 2016-12-14 18:21 - Jing Tao

This is my feeling as well.

#4 - 2017-01-11 21:10 - Jing Tao

- Assignee changed from Jing Tao to Rob Nahf

#5 - 2017-01-11 23:47 - Rob Nahf

- Parent task set to #7933

#6 - 2017-01-12 00:00 - Rob Nahf

- Status changed from New to In Progress

- % Done changed from 0 to 30

stopping the index processor is done via services d1-index-processor stop command (/etc/init.d/d1-index-processor stop) on the command line.

The stop command successfully stops the scheduler but isn't interrupting the ExecutorService that has spooled up an in-memory queue of tasks to execute (by 5 worker threads).

The IndexTaskScheduler.stop() method needs to send an interrupt to each executing IndexTaskProcessorJob, which in turn needs to tell its IndexTaskProcessor to finish fast. This means Calling shutdown on the ExecutorService. shutdown cancels all of the queued tasks it hasn't gotten to yet, guaranteeing that the worker threads only finish up their current task.

Those canceled tasks will have to be put back into NEW state, either at shutdown, or at next start up.

#7 - 2017-01-12 05:49 - Rob Nahf

- Subject changed from D1-index-processor should be shut down soon to D1-index-processor should be shut down soon after stop given

#8 - 2017-01-26 16:36 - Rob Nahf

- % Done changed from 30 to 50

- Status changed from In Progress to Testing

#9 - 2017-02-02 17:46 - Rob Nahf

- Description updated

testing of 2.3(.1) branch in SANDBOX shows quick shutdown, but failure to move canceled tasks out of 'IN PROCESS' state and back to 'NEW'. the related logging statements also are not being logged, leading me to believe that there may be some class loader involvement with the scheduler, because the method which implements shutdown is a static method.

IF immediate safe shutdown is affected, although tasks left in bad status, that could be good enough for the point release.

#10 - 2017-02-10 23:51 - Rob Nahf

Finally got shutdown to work gracefully and clean up after itself. Not the cleanest looking code. Creating such a long execution queue is considered a bad practice. Better would be to have the processor periodically top-off a limited blocking queue when the number of tasks is getting low. (see http://www.nurkiewicz.com/2014/11/executorservice-10-tips-and-tricks.html)

#11 - 2017-02-10 23:51 - Rob Nahf

- Status changed from Testing to Closed

- % Done changed from 50 to 100