Item 83
git_comments:

1. nocommit: need to raise a LUCENE jira for this?
2. Any additional auxiliary data that needs to be stored
3. is this better here, or on SolrQueryRequest?
4. unset
5. nocommit
6. TODO: if docset is not needed, avoid this initialization
7. nocommit: need docset should be false

git_commits:

1. **summary:** SOLR-13350: Multi-threaded search through an index
   message: SOLR-13350: Multi-threaded search through an index

github_issues:

github_issues_comments:

github_pulls:

github_pulls_comments:

github_pulls_reviews:

jira_issues:

1. **summary:** Explore collector managers for multi-threaded search
   description: AFAICT, SolrIndexSearcher can be used only to search all the segments of an index in series. However, using CollectorManagers, segments can be searched concurrently and result in reduced latency. Opening this issue to explore the effectiveness of using CollectorManagers in SolrIndexSearcher from latency and throughput perspective.
2. **summary:** Explore collector managers for multi-threaded search
   description: AFAICT, SolrIndexSearcher can be used only to search all the segments of an index in series. However, using CollectorManagers, segments can be searched concurrently and result in reduced latency. Opening this issue to explore the effectiveness of using CollectorManagers in SolrIndexSearcher from latency and throughput perspective.
3. **summary:** Explore collector managers for multi-threaded search
   description: AFAICT, SolrIndexSearcher can be used only to search all the segments of an index in series. However, using CollectorManagers, segments can be searched concurrently and result in reduced latency. Opening this issue to explore the effectiveness of using CollectorManagers in SolrIndexSearcher from latency and throughput perspective.
   label: code-design
4. **summary:** Explore collector managers for multi-threaded search
   description: AFAICT, SolrIndexSearcher can be used only to search all the segments of an index in series. However, using CollectorManagers, segments can be searched concurrently and result in reduced latency. Opening this issue to explore the effectiveness of using CollectorManagers in SolrIndexSearcher from latency and throughput perspective.
5. **summary:** Explore collector managers for multi-threaded search
   description: AFAICT, SolrIndexSearcher can be used only to search all the segments of an index in series. However, using CollectorManagers, segments can be searched concurrently and result in reduced latency. Opening this issue to explore the effectiveness of using CollectorManagers in SolrIndexSearcher from latency and throughput perspective.
   label: code-design
6. **summary:** Explore collector managers for multi-threaded search
   description: AFAICT, SolrIndexSearcher can be used only to search all the segments of an index in series. However, using CollectorManagers, segments can be searched concurrently and result in reduced

latency. Opening this issue to explore the effectiveness of using CollectorManagers in SolrIndexSearcher from latency and throughput perspective.
7. **summary:** Explore collector managers for multi-threaded search
   description: AFAICT, SolrIndexSearcher can be used only to search all the segments of an index in series. However, using CollectorManagers, segments can be searched concurrently and result in reduced latency. Opening this issue to explore the effectiveness of using CollectorManagers in SolrIndexSearcher from latency and throughput perspective.
8. **summary:** Explore collector managers for multi-threaded search
   description: AFAICT, SolrIndexSearcher can be used only to search all the segments of an index in series. However, using CollectorManagers, segments can be searched concurrently and result in reduced latency. Opening this issue to explore the effectiveness of using CollectorManagers in SolrIndexSearcher from latency and throughput perspective.
   label: code-design
9. **summary:** Explore collector managers for multi-threaded search
   description: AFAICT, SolrIndexSearcher can be used only to search all the segments of an index in series. However, using CollectorManagers, segments can be searched concurrently and result in reduced latency. Opening this issue to explore the effectiveness of using CollectorManagers in SolrIndexSearcher from latency and throughput perspective.
10. **summary:** Explore collector managers for multi-threaded search
    description: AFAICT, SolrIndexSearcher can be used only to search all the segments of an index in series. However, using CollectorManagers, segments can be searched concurrently and result in reduced latency. Opening this issue to explore the effectiveness of using CollectorManagers in SolrIndexSearcher from latency and throughput perspective.
11. **summary:** Explore collector managers for multi-threaded search
    description: AFAICT, SolrIndexSearcher can be used only to search all the segments of an index in series. However, using CollectorManagers, segments can be searched concurrently and result in reduced latency. Opening this issue to explore the effectiveness of using CollectorManagers in SolrIndexSearcher from latency and throughput perspective.
12. **summary:** Explore collector managers for multi-threaded search
    description: AFAICT, SolrIndexSearcher can be used only to search all the segments of an index in series. However, using CollectorManagers, segments can be searched concurrently and result in reduced latency. Opening this issue to explore the effectiveness of using CollectorManagers in SolrIndexSearcher from latency and throughput perspective.
13. **summary:** Explore collector managers for multi-threaded search
    description: AFAICT, SolrIndexSearcher can be used only to search all the segments of an index in series. However, using CollectorManagers, segments can be searched concurrently and result in reduced latency. Opening this issue to explore the effectiveness of using CollectorManagers in SolrIndexSearcher from latency and throughput perspective.

jira_issues_comments:

1. Extremely early, barely functional patch (as per Yonik's law of patches). Need quite a few iterations from here on. Attached it so that if I drop the ball, someone else can pick it up from here.
2. Updating the patch with more functionality covered. Enabled this by default (as opposed to enabling via a parameter, which is what we want to do), and all tests passing. Still, lots of cleanup and refactoring pending.
3. **body:** Updated patch with the following: # Added an executor service (which is used for the multi-threaded search). # TODO: JoinQParserPlugin closes the "from core" on every thread of the search (using a close hook). This causes problems with multiple core close calls for the same request. Tried to handle it naively for now (to make the tests pass). Does someone have ideas to tackle it properly? # TODO: Add a parameter to selectively enable multi-threaded search. Use that parameter randomly with tests. Right now, except the Join tests (esp. TestCrossCoreJoin), all pass.
   label: code-design
4. Opened https://github.com/apache/lucene-solr/pull/675 in GitHub for easier review. From the last time, here are the changes: 1. Fixed the cross core join tests. In current Solr (single threaded search), a "from core" is created during the constructor of the weight class, and the core is closed using the SolrRequestInfo's close hook. In multi-threaded search (this jira), the core was being closed when every thread's SolrRequestInfo#clearRequestInfo() is called (which in turn calls the close hook). Hence, there were multiple close calls on the same core reference. In this patch, the "from core" is now created once

per thread (in a newly introduced initHook in SolrRequestInfo). 2. TODO: Add a parameter to selectively enable multi-threaded search. Use that parameter randomly with tests.

5. **body:** In general, it seems like an executor for parallel searches would be more useful at the CoreContainer level. If the executor is per-searcher, then picking a high enough pool size for good concurrency for a single core means that one would get way to many threads if one has tons of cores per node (not that unusual) We should also audit all Weight classes in Solr for thread safety (if it hasn't been done yet.) . Relying on existing tests to catch stuff like that won't work that well for catching race conditions.
   label: code-design
6. I just want to point out that this idea was explored by Etsy who presented their findings in a Lucene Revolution presentation years ago: [https://www.slideshare.net/lucidworks/searchtime-parallelism-presented-by-shikhar-bhushan-etsy-41862845] One issue that I remember is that there is difficulty in concurrently using a DocSetCollector (subsequently used for filter cache and stats/facets) which was not designed for concurrency. You can see the slides cover that.
7. -FYI, I'm trying to update the patch to latest master. It is taking me longer due to changes in SOLR-13892.- Added a new PR for this (as the earlier one fell completely out of date): https://github.com/apache/lucene-solr/pull/1310 Would appreciate if someone can please review.
8. **body:** bq. One issue that I remember is that there is difficulty in concurrently using a DocSetCollector Thanks [~dsmiley]. Here, I've not used a shared DocSetCollector, but every thread uses its own, and I'm intersecting all of those at the end of the search. This might not be as performant, but perhaps good for a start.
   label: code-design
9. As per an offline conversation with [~atris], I requested him to revive this issue and work on taking it to the completion. I will help him as he does so. Thanks for your help, [~atris]. FYI, Atri has optimized this feature at Lucene level and has intimate knowledge of how it works internally.
10. I have been bringing this up to date and fixing some outstanding comments at: [https://github.com/atris/lucene-solr/tree/solr-13350]
11. Commit 36f268b65b12bb05da700f0a2c843acca7f30af5 in lucene-solr's branch refs/heads/tmp from Ishan Chattopadhyaya [https://gitbox.apache.org/repos/asf?p=lucene-solr.git;h=36f268b] SOLR-13350: Multi-threaded search using collectors manager