WritingDistributedSearchComponents

Basics

⚠️ Note: Best efforts are being made to make this documentation reliable, but please consider it a work in progress. ⚠️

A Distributed SearchComponent is a SearchComponent that overrides one or more of the following methods:

1. distributedProcess
2. modifyRequest
3. handleResponses
4. finishStage

The distributed search process takes the place of the process() API to the SearchComponent. The SearchHandler first calls the prepare() functions on all of the components. If a component needs to act differently in prepare() in the distributed case, it can check RequestBuilder.isDistrib. This boolean will be true for a request that is about to be distributed. (Also, params contains ShardParams.SHARDS at top level here.) The prepare() and perform() API will be called out on the shards to perform each shard's worth of the query.

There are 4 stages to distributed search:

1. Start (ResponseBuilder.STAGE_START) 2. Query Parse (ResponseBuilder.STAGE_PARSE_QUERY) 3. Execute Query (ResponseBuilder. STAGE_EXECUTE_QUERY) 4. Get Fields (ResponseBuilder.STAGE_GET_FIELDS)

(There's also ResponseBuilder.STAGE_DONE, which should be returned when the component has nothing left to do.)

The basic distributed algorithm (which is a different path from the non-distributed), as implemented in SearchHandler is:

1. while not at STAGE_DONE
    a. For each Component invoke distributedProcess to see if it has anything it wants to distribute. If it does, it will return one of the 4 stages and setup a ShardRequest and add it to a queue
        i. modifyRequest() can be used to refine other shard requests, instead of sending a separate ShardRequest. See the FacetComponent for an example. This is likely best done by calling ResponseBuilder.addRequest() which will then invoke the modifyRequest() method on each of the components.
        ii. Components can also indicate the purpose of their request by using the ShardRequest.PURPOSE_* fields. When set, these fields can then be inspected by the other components to see if it is a purpose they are interested in.
    b. While we have outgoing ShardRequest instances
        i. Submit the requests to the shards. The QT for these requests will be null (the default) unless the original query, or one of the components, set the the ShardParams.SHARDS_QT parameter to something else. The ShardParams.SHARD_URL passes along the original URL to the shards.
        ii. For each component, handle the responses
    c. For each component, call finishStage()

In this process, each ShardRequest then goes off to it's Shard just as if it were a normal request of that server.

Testing

org.apache.solr.BaseDistributedSearchTestCase provides test support for distributed components. It compares the results of distributed and non-distributed queries.

Classes to Know

QueryComponent

If you are writing a search component that you add to the standard list, rather than replacing the usual suspects, you will need to work with the behavior of the core component: QueryComponent.

1. in prepare(), QueryComponent parses the query, storing all of the resulting information in the ResponseBuilder.
   2. In distributedProcess(), QueryComponent forks down two different paths depending on whether the query is grouped or not. For non-grouped queries, it performs two interactions with the shards. The first sends the original query out, but with the field list trimmed down to the document ID and the score. For a grouped query, an explanation is *to be written*. 3. in handleResponses, for a regular query, the results are merged, and then QueryComponent sends out the second query, to retrieve the desired fields from the original FL. When these responses return, the final results are packaged up into the ResponseBuilder.

ResponseBuilder

Key distributed member variables are:

1. shards - The address of the shards, not including http://
2. stage - What stage this response is in, out of the 5 stages specified above.
3. isDistrib - true in the distribution algorithm, false out on the shards.

ShardRequest

⚠️ :TODO: ⚠️

ShardResponse

⚠️ :TODO: ⚠️

⚠️ :TODO: ⚠️ Fill in when to override what. Fill in how to know what stage your at. Fill in how to signify the component is done with its work