<http://stackoverflow.com/questions/25884598/when-to-use-cassandra-vs-solr-in-dse>

|  |  |  |
| --- | --- | --- |
| I'm using DSE for Cassandra/Solr integration so that data are stored in Cassandra and indexed in Solr. It's very natural to use Cassandra to handle CRUD operation and use Solr for full text search respectively, and DSE can really simplify data synchronization between Cassandra and Solr.  When it comes to query, however, there are actually two ways to go: Cassandra secondary/manual configured index vs. Solr. I want to know when to use which method and what's the performance difference in general, especially under DSE setup.  Here is one example use case in my project. I have a Cassandra table storing some item entity data. Besides the basic CRUD operation, I also need to retrieve items by equality on some field (say category) and then sort by some order (in my case here, a like\_count field).  I can think of three different ways to handle it:   1. Declare 'indexed=true' in Solr schema for both category and like\_count field and query in Solr 2. Create a denormalized table in Cassandra with primary key (category, like\_count, id) 3. Create a denormalized table in Cassandra with primary key (category, order, id) and use an external component, such as Spark/Storm，to sort the items by like\_count   The first method seems to be the simplest to implement and maintain. I just write some trivial Solr accessing code and the rest heavy lifting are handled by Solr/DSE search.  The second method requires manual denormalization on create and update. I also need to maintain a separate table. There is also tombstone issue as the like\_count can possibly be updated frequently. The good part is that the read may be faster (if there are no excessive tombstones).  The third method can alleviate the tombstone issue at the cost of one extra component for sorting.  Which method do you think is the best option? What is the difference in performance?  [solr](http://stackoverflow.com/questions/tagged/solr) [cassandra](http://stackoverflow.com/questions/tagged/cassandra) [datastax-enterprise](http://stackoverflow.com/questions/tagged/datastax-enterprise)   |  |  | | --- | --- | | [share](http://stackoverflow.com/q/25884598)|[improve this question](http://stackoverflow.com/posts/25884598/edit) | asked Sep 17 '14 at 7:19  [[https://www.gravatar.com/avatar/df9ee2f42bb8dfa38683ec3ab2d4a647?s=32&d=identicon&r=PG&f=1](http://stackoverflow.com/users/3988464/ziju-feng)](http://stackoverflow.com/users/3988464/ziju-feng)  [Ziju Feng](http://stackoverflow.com/users/3988464/ziju-feng) **37**3 | |
|  | add a comment |

**1 Answer**

[active](http://stackoverflow.com/questions/25884598/when-to-use-cassandra-vs-solr-in-dse?answertab=active#tab-top)[oldest](http://stackoverflow.com/questions/25884598/when-to-use-cassandra-vs-solr-in-dse?answertab=oldest#tab-top)[votes](http://stackoverflow.com/questions/25884598/when-to-use-cassandra-vs-solr-in-dse?answertab=votes#tab-top)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| up vote5down voteaccepted | Cassandra secondary indexes have limited use cases:   1. No more than a couple of columns indexed. 2. Only a single indexed column in a query. 3. Too much inter-node traffic for high cardinality data (relatively unique column values) 4. Too much inter-node traffic for low cardinality data (high percentage of rows will match) 5. Queries need to be known in advance so data model can be optimized around them.   Because of these limitations, it is common for apps to create "index tables" which are indexed by whatever column is desired. This requires either that data be duplicated from the main table to each index table, or an extra query will be needed to read the index table and then read the actual row from the main table after reading the main key from the index table. Queries on multiple columns will have to be manually indexed in advance, making ad hoc queries problematic. And any duplicated will have to be manually updated by the app into each index table.  Other than that... they will work fine in cases where a "modest" number of rows will be selected from a modest number of nodes, and queries are well specified in advance and not ad hoc.  DSE/Solr is better for:   1. A moderate number of columns are indexed. 2. Complex queries with a number of columns/fields referenced - Lucene matches all specified fields in a query in parallel. Lucene indexes the data on each node, so nodes query in parallel. 3. Ad hoc queries in general, where the precise queries are not known in advance. 4. Rich text queries such as keyword search, wildcard, fuzzy/like, range, inequality.   There is a performance and capacity cost to using Solr indexing, so a proof of concept implementation is recommended to evaluate how much additional RAM, storage, and nodes are needed, which depends on how many columns you index, the amount of text indexed, and any text filtering complexity (e.g., n-grams need more.) It could range from 25% increase for a relatively small number of indexed columns to 100% if all columns are indexed. Also, you need to have enough nodes so that the per-node Solr index fits in RAM or mostly in RAM if using SSD. And vnodes are not currently recommended for Solr data centers.   |  |  |  | | --- | --- | --- | | [share](http://stackoverflow.com/a/25892212)|[improve this answer](http://stackoverflow.com/posts/25892212/edit) | [edited Sep 18 '14 at 3:55](http://stackoverflow.com/posts/25892212/revisions) | answered Sep 17 '14 at 13:39  [[http://i.stack.imgur.com/AtwiZ.jpg?s=32&g=1](http://stackoverflow.com/users/2769371/jack-krupansky)](http://stackoverflow.com/users/2769371/jack-krupansky)  [Jack Krupansky](http://stackoverflow.com/users/2769371/jack-krupansky) **661**14 | |
|  | |  |  |  |  | | --- | --- | --- | --- | | |  |  | | --- | --- | |  |  | | +1 Brilliant answer. And I totally agree with secondary indexes having limited use cases. Probably the most mis-understood tool in Cassandra right now. –  [BryceAtNetwork23](http://stackoverflow.com/users/1054558/bryceatnetwork23) [Sep 17 '14 at 13:56](http://stackoverflow.com/questions/25884598/when-to-use-cassandra-vs-solr-in-dse#comment40523563_25892212) | | |  |  | | --- | --- | |  |  | | +1 I couldn't have said it any better. I recently came across this dilemma too and found myself using Solr for ALL read operations because Cassandra couldn't filter on more than one column per query (basically, because Cassandra secondary indexes can only be declared on one column at a time - i.e. there are no compound indexes). For me, this is the main limitation. –  [PJ.](http://stackoverflow.com/users/858481/pj) [Sep 18 '14 at 9:16](http://stackoverflow.com/questions/25884598/when-to-use-cassandra-vs-solr-in-dse#comment40552951_25892212) | |