SpatialClustering

So you've got spatial data, lets say just points, and you want to plot it on a map. Plotting only the top results (corresponding with the 'rows' param) is one thing but what about plotting all points that match the query? Sure, you could set 'rows' to a super-large value but that will be *slow*. high 'rows' values is a Solr anti-pattern from a performance perspective, plus your mapping toolkit is going to be another bottleneck if it's asked to plot too many points.

One way to look at this problem is grid based heatmap clustering. All points in a grid square get counted to give a grid square a numeric value, and those values correspond to a color scale.

Heatmaps

Solr 5.1 Heatmaps:

Solr 5.1 has heatmaps, a type of faceting. See the Solr Reference Guide, which should have information on how to use it by the time 5.1 ships. There's also the JIRA issue: https://issues.apache.org/jira/browse/SOLR-7005

Before Solr 5.1:

What follows is an (edited) email from David Smiley to some folks on this subject:

The stackoverflow response that I wrote that is apt is this: http://stackoverflow.com/questions/11319465/geoclusters-in-solr/11321723#11321723

Possible adjustments to the idea:

- put each geohash length into a separate field (e.g. geohash_1😃 geohash_2:DR, ...) and then each field could be single-valued assuming you only have one thing to plot per document
- you could skip geohash_1 since presumably you'd never want a grid that course, not even for the world. And likewise only go down to the length where this approach is useful – after which you might simply always plot the points instead of a heatmap because you know there would never be that many points.
- Instead of faceting only for the geohash length, such as facet.prefix=2_ (or in the case of the above, using a dedicated field for the length), you could also add some leading geohash characters that you know limit the scope of where the data is for your current filter. To do this, you could use some of the existing Lucene spatial code – SpatialPrefixTree which will return a set of cells corresponding to prefixes at a desired resolution (pretty course for this scenario – you only want a half dozen maybe). Then for each of these cells, you generate a separate facet.field query with a corresponding prefix. Yes you can facet on the same field multiple ways using the 'key' local-param. The point of all this in this bullet is to make it faster, so don't bother if it's fast enough.

So you want to plot points using different icons based on a small set of categories? Amend the previous strategy by appending a character at the end of the geohash (for each length) to designate which of your categories it's for. To save memory, keep it to a character that you can lookup. For example geohash_2:DR0 (the '0' was arbitrarily added and signifies some particular category). Then when you get the facet results back, for a given cell like "DR" you have the facets for each category.

I once regrettably referred to the approach above as the "poor man's" approach but it really isn't bad at all. It's something that can be done without hacking Lucene/Solr itself.

A custom Solr component could be developed to do these things to make it a little faster and definitely easy to use by the Solr client, and re-usable too. A specific optimization it could do to reduce the response size is to use a grid (matrix) of numbers instead of returning a geohash for each cell and forcing the client to decode it. As you try to increase the resolution of the grid to make it look nicer, this will become an issue. Another thing it could do is make the heatmap dynamic. Instead of returning a cell with a count, return the actual points at precise locations if there aren't that many on a cell-by-cell basis.

What I wonder about is memory usage. This approach will require a fair amount of RAM by default. There are different ways of tuning Solr (especially Solr 4.2) that will have different RAM and faceting performance characteristics, all via simple configuration (usually re-indexing or at least using different fields for different experiments).

Misc

See this YouTube video presentation at DrupalCon: http://s.apache.org/DrupalConSpatial

One of the techniques used was to use the stats component to compute an average lat/lon of all points in a geohash grid. You would do stats.facet on the geoghash grid of suitable length, and compute stats.field on latitude as well as longitude.