Geostreams Caching

Current Implementation

The current implementation is enabled by setting the geostream.cache to a path on disk where the files are stored. Calls to api.Geostreams.binDatapoints (...) and api.Geostreams.searchDatapoints(...) will use cacheFetch() to retrieve the response from the cache if available or create it, put it in the cache and then return it. The files on disk are stored by hashing the request. Two files are stored on disk under the hash, the response as a json text file (without spaces) and a .json file that includes the actual request as json (for inspection of the raw files). For example:

```
lmarini@seagrant-dev:/tmp/clowder$ ls
0942B7099D8A24351E0855A0792B716A       4803BDC1330BDB0B169A0A0CCB8D924B.json  B931A28FCB2740782FE7133CE8E59920
0942B7099D8A24351E0855A0792B716A.json  4D972E3DB7D37CEC47A98A633666CA84       B931A28FCB2740782FE7133CE8E59920.json
09E7B84AB3577182197094ECBF51618B       4D972E3DB7D37CEC47A98A633666CA84.json  BAE8CBEB00310003693BE4B2D96FA65F
09E7B84AB3577182197094ECBF51618B.json  53C8BE76B300BB64CB5F456C33ED32FE       BAE8CBEB00310003693BE4B2D96FA65F.json
0BEDDDCB40472E7DED5BC4DACFD5F492       53C8BE76B300BB64CB5F456C33ED32FE.json  BBC3590245AC84AF0425B4F51330B05E
0BEDDDCB40472E7DED5BC4DACFD5F492.json  574C35E8FF4B4763FB20A4D5E6728947       BBC3590245AC84AF0425B4F51330B05E.json
0DAC61DBF8042941496DB397D50F1D41       574C35E8FF4B4763FB20A4D5E6728947.json  C6191159810DA9B8656A85573295A7DA
0DAC61DBF8042941496DB397D50F1D41.json  604ADA6A548061BB7A0116C84CD7D2CD       C6191159810DA9B8656A85573295A7DA.json
0E6AED83B420A0107B120FFD1E6F16DE       604ADA6A548061BB7A0116C84CD7D2CD.json  C64ED8689D9FA445009C8D2D8864BAEC
0E6AED83B420A0107B120FFD1E6F16DE.json  6B6B1D983B04935CCEE9B3AACCEDC7F2       C64ED8689D9FA445009C8D2D8864BAEC.json
0E94BA94F136E5A4870FAD6B5BBE295D       6B6B1D983B04935CCEE9B3AACCEDC7F2.json  CEDEE899ABE3D900606971B9026B0730
0E94BA94F136E5A4870FAD6B5BBE295D.json  7D3F43A0B4FA2064D5E87697AFF0B27E       CEDEE899ABE3D900606971B9026B0730.json
11B41F30307468F53C50A422D7FCF1C4       7D3F43A0B4FA2064D5E87697AFF0B27E.json  D2BBA6D84C33C01C3FD5FA73FFBFCB61
11B41F30307468F53C50A422D7FCF1C4.json  7DEAFD92D9976AA3A42D7BE8F1BD4A32       D2BBA6D84C33C01C3FD5FA73FFBFCB61.json
14426267CEFC2724F2B1AA57675DB886       7DEAFD92D9976AA3A42D7BE8F1BD4A32.json  D4878B444C8343615DDADD11881B73C1
14426267CEFC2724F2B1AA57675DB886.json  7EE1D7C966EB8DD0B4F32CC11EE3C4B0       D4878B444C8343615DDADD11881B73C1.json
173610A9507BEFFF056557A1A5FC7EF1       7EE1D7C966EB8DD0B4F32CC11EE3C4B0.json  D509773CEB1C2F5C7CAC58AE6BC04065
173610A9507BEFFF056557A1A5FC7EF1.json  81B82381AE471438AB6C524B58B18505       D509773CEB1C2F5C7CAC58AE6BC04065.json
1B3536F1E7862A8F7C61613589BD6D12       81B82381AE471438AB6C524B58B18505.json  D528474AD4829BB29B7B6683929209F3
1B3536F1E7862A8F7C61613589BD6D12.json  8C68E6E25F0CEE35A3E5F1A974B7746C       D528474AD4829BB29B7B6683929209F3.json
```

Here are two example queries from a .json file:

```
{"format":"json","operator":"","since":"","until":"","geocode":"","stream_id":"7196","sensor_id":"","sources":
[],"attributes":[],"semi":""}

{"time":"season","depth":1.0,"since":"","until":"","geocode":"","stream_id":"","sensor_id":"844","sources":[],"
attributes":[]}
```

Here is an example of the response:

```
{"sensor_name":"grid0713","properties":{"species":[],"toxaphene_var":[
{"depth":0.0,"label":"1991 winter","sources":["http://seagrant-dev.ncsa.illinois.edu/medici/datasets
/54591ee1e4b0e9dff1baf336"],"year":1991,"date":"1991-02-01T12:00:00.000-06:00","depth_code":"NA","average":
0.0477260702109212,"count":1},
{"depth":0.0,"label":"1991 spring","sources":["http://seagrant-dev.ncsa.illinois.edu/medici/datasets
/54591ee1e4b0e9dff1baf336"],"year":1991,"date":"1991-05-01T12:00:00.000-05:00","depth_code":"NA","average":
0.0477260702109212,"count":1},
{"depth":0.0,"label":"1991 summer","sources":["http://seagrant-dev.ncsa.illinois.edu/medici/datasets
/54591ee1e4b0e9dff1baf336"],"year":1991,"date":"1991-08-01T12:00:00.000-05:00","depth_code":"NA","average":
0.0477260702109212,"count":1},
{"depth":0.0,"label":"1991 fall","sources":["http://seagrant-dev.ncsa.illinois.edu/medici/datasets
/54591ee1e4b0e9dff1baf336"],"year":1991,"date":"1991-11-01T12:00:00.000-06:00","depth_code":"NA","average":
0.0477260702109212,"count":1},
{"depth":0.0,"label":"1993 winter","sources":["http://seagrant-dev.ncsa.illinois.edu/medici/datasets
/54591ee1e4b0e9dff1baf336"],"year":1993,"date":"1993-02-01T12:00:00.000-06:00","depth_code":"NA","average":
0.139673763930734,"count":1},
{"depth":0.0,"label":"1993 spring","sources":["http://seagrant-dev.ncsa.illinois.edu/medici/datasets
/54591ee1e4b0e9dff1baf336"],"year":1993,"date":"1993-05-01T12:00:00.000-05:00","depth_code":"NA","average":
0.139673763930734,"count":1},
{"depth":0.0,"label":"1993 summer","sources":["http://seagrant-dev.ncsa.illinois.edu/medici/datasets
/54591ee1e4b0e9dff1baf336"],"year":1993,"date":"1993-08-01T12:00:00.000-05:00","depth_code":"NA","average":
0.139673763930734,"count":1}
```

Current potential issues:

1. Lots of files are created on disk

2. Files on disk don't seem to include since and until, result in potentially more data being sent to the client
3. Admin has to prime the cache (sometimes we forget)

Proposed Implementation

1. Move the cache to postgresql.
2. Each aggregate datapoint could be a row in a table so that queries could be more specific to a certain range / sensor / stream.
3. Keep aggregations (yearly, semi, seasonal, monthly, daily, hourly) in separate tables.
    a. For example bins_year would include all yearly averages.
    b. Columns could be (id:int:, sensor:int, stream:int, year:int, data:json, averages:json).
        i. The data column could store the current total and count for each variable updating running averages.
        ii. The averages column could store the current average. This way returning the actual values will not require any further computation.
4. Each new added datapoint triggers updates on the aggregations tables. This will only update one row per table.