Globals for the Environment

v 1.1

the HotSpots Edition

by William Cheung
Toronto, April 5, 2012

Introduction

For the previous challenge of demonstrating innovative use of (big) data, I used Globals to build GGSMR, the *Globals Big Geospatial Data MapReducer*, a desktop app which I applied to map/reduce 5.2 million ocean CO2 data points from 1957-2011, visualizing the results in a web app, *Globals for the Environment*. For the current challenge I am presenting the *GGSMR Hotspot Finder*, a command line program which uses Globals to post-process the map/reduce output of GGSMR to find the top 10 ocean CO2 “hotspots,” visualizing the results as a “heat map” in an enhanced version of *Globals for the Environment*.

The Problem Identified

Recall that the original *Globals for the Environment* visualization (web app) presents a world map where you can click a region to see a graph of ocean surface CO2 levels in the region over the last 50+ years. While this interface provides an environmental researcher with the ability to drilldown and see the trend in a particular region, it does not provide any global view useful to raise alerts of danger zones. Global warming visualizations typically have a concept of a heat map, and I decided to add this feature to *Globals for the Environment* so that a researcher could see at a glance the top 10 hotspots, regions of high recent CO2 levels on the global oceans.

The Solution

The map/reduce algorithm, such as implemented in GGSMR, does not have the concept of ordering, which is required to determine a top 10 list. For the ocean CO2 dataset, ordering is obviously required to compare CO2 levels in different regions, to rank each region. Less obviously, ordering is also required in each region, to determine the most recent year for which data is available. That's because not all years have data available, say if a research ship has not passed through the region in a long time. A researcher should have the ability to exclude a region which has "too old" data from the top 10 list.

Globals is ideal for ordering computations because subscripts at the same level in a global are always ordered. For the problem solution, I leveraged this feature to build the *GGSMR Hotspot Finder*, a command line program written in Groovy which queries the global built by GGSMR and outputs a top average rankings list to both the console and a

CSV file. It is the CSV file which is used by another Groovy program *Top10OceanCO2Highlighter* to create a heat map for *Globals for the Environment*.

GGSMR Hotspot Finder Algorithm

Recall that the final output of the MapReducer job of GGSMR is a global consisting of leaf nodes with subscripts *[latUp, latDown, lonLeft, lonRight, year]* and value of CSV list: *samples, min, max, average*. For example, a node has subscripts [-60,-90, -60,-30, 2010] and value: 12206, 172.8, 459.2, 401.0949943.

GGSMR Hotspot Finder first navigates through this global to collect all the region subscripts, for example [-60,-90, -60,-30]. Essentially it navigates to a maximum node depth of 4 because a region has 4 geospatial co-ordinates represented as subscripts. Then with all region subscripts collected, it iterates through this set of subscripts to call previousSubscript on the 5th node level in the global (the 5th subscript represents the year). For each region it only needs to call previousSubscript once with the maximum possible integer (representing the end of time), and the result is the most recent year in the region with data (CO2 data in our scenario). Now that it has each region along with the most recent data year for the region, the Hotspot Finder simply has to get the node data in the global at those 5 subscripts.

Remember that the node data is a CSV list: *samples, min, max, average*. Hotspot Finder parses the CSV list to extract the average as a double and uses the average as the first level subscript in a temporary, scratch-pad global it creates. The data in the scratch-pad global is a CSV list of the other info associated with this average, namely: *latUp, latDown, lonLeft, lonRight, year*. For example, the leaf node described above is represented in the scratch-pad global with subscript [401.0949943] and data: -60,-90, -60,-30, 2010. By using average as a subscript, Hotspot Finder can leverage subscript ordering to make 10 calls to previousSubscript to get the top 10 averages (passing the maximum possible double in the first previousSubscript call). For each average, it gets the node data consisting of the region and most recent year for the average.

A detail of this determination of the top 10 averages is that Hotspot Finder introduces a “duplicate resolver” subscript in case different regions have the same most recent average. So for the example, the leaf node in the scratch-pad global actually has subscripts like [401.0949943, 5], where the 5 subscript value is a simple counter representing this is the 5th average obtained.

Another detail of Hotspot Finder is the user can specify how far back they want to go for including region data in the rankings. For example with the ocean CO2 dataset which covers years 1957–2011, the user can specify to only go back one year from 2011, so that data earlier than 2010 is not included in determining a current top 10 list.

GGSMR Hotspot Finder Example Run

The *GGSMR Hotspot Finder* is designed to find hotspots in the map/reduce output of GGSMR the *Globals Big Geospatial Data MapReducer*. Both *GGSMR Hotspot Finder* and GGSMR are designed to work with an arbitrary geospatial dataset contained in a single text file of any size. The example will use the *LDEO Ocean CO2 dataset* referenced in my previous challenge submission.

<u>Steps</u>

1. Download the *LDEO ocean CO2 dataset* and unzip it:
http://cdiac.ornl.gov/ftp/oceans/LDEO_Database/Version_2010/LDEO_Database_V2010.txt.tar.gz
Warning: the unzipped file is 543 megabytes consisting of 5,276,054 lines.

2. Follow the instructions from my previous challenge submission PDF to run GGSMR to map/reduce the LDEO dataset. When the map/reduce is finished, do not delete the reducer output global.

Essentially in the src directory run:

groovy Main

Select the unzipped LDEO file for "geospatial dataset file" and click "map reduce."

3. In the src directory run the *GGSMR Hotspot Finder* help option:
groovy GGSMRHotspotFinder -h

This displays the options available and their defaults:

```
usage: groovy GGSMRHotspotFinder -r reducerOutputGlobal -t 10 -l 2011 -m 1
              -o /top10AvgRankings.csv
 -h,--help                               usage
 -l,--latestYearWithData <arg>           default: 2011
 -m,--maxYearsToGoBackFromLatest <arg>   default: 1
 -o,--outputFile <arg>                   default: /top10AvgRankings.csv
 -r,--reducerOutputGlobal <arg>          default:
                                         reducerOutput.yearly.30x30
 -t,--top <arg>                          default: 10
 -v,--verbose                            verbose output
```

4. Assuming the defaults are acceptable, run:
groovy GGSMRHotspotFinder

This computes and displays the top 10 rankings:

```
rank 1  0,-30,-120,-90,2010,448.45143005720251494
rank 2  30,0,150,180,2010,410.74767975522757978
rank 3  -60,-90,-60,-30,2010,401.09499426511843012
rank 4  30,0,-60,-30,2010,400.89539682539663091
rank 5  60,30,150,180,2010,395.73076369394590301
rank 6  30,0,-90,-60,2010,395.41964348129960171
rank 7  60,30,120,150,2010,390.11569926873852409
rank 8  0,-30,-150,-120,2010,388.90621800662506757
rank 9  30,0,-120,-90,2010,386.77858002406748028
rank 10 0,-30,-180,-150,2010,380.24474576271137493
```

By default the data (the output without the rank labels) is saved in file *top10AvgRankings.csv* in the root directory. If this is not appropriate, use the -o option (or --outputFile) to specify an alternate file path and name. If the output file already exists, Hotspot Finder will warn you:

```
Warning: File C:\top10AvgRankings.csv will be cleared. Hit Enter to continue...
```

(Although this example message shows a DOS file path, the programs run on any O/S supporting Java.)

Note the default name of the reducer output global used is the same as the default in GGSMR: *reducerOutput.yearly.30x30*. If you used another name in GGSMR, use the -r option (or --reducerOutputGlobal) to specify the same name to Hotspot Finder.

5. By default the top 10 rankings are computed. If you want the top 15 say, then run:
groovy GGSMRHotspotFinder -t 15

6. For verbose output, use the -v option:
groovy GGSMRHotspotFinder -t 15 -v

```
region 1        -60,-90,-180,-150,2011  235.85000000000008
region 2        -60,-90,-150,-120,2011  363.85279720279715
region 3        -60,-90,-120,-90,2011   330.98597243491463
region 4        -60,-90,-90,-60,2010    343.7160496311179
region 5        -60,-90,-60,-30,2010    401.09499426511843
region 6        -60,-90,-30,0,2010      336.70974529485557
region 7        -60,-90,0,30,2010       322.23972589792083
region 8        -60,-90,30,60,2008      384.5652675886952       SKIPPED 2008 too far back
region 9        -60,-90,60,90,2007      314.4672146596857       SKIPPED 2007 too far back
region 10       -60,-90,90,120,2007     298.4983060109288       SKIPPED 2007 too far back
region 11       -60,-90,120,150,2007    340.7423106796115       SKIPPED 2007 too far back
region 12       -60,-90,150,180,2008    325.1223297654284       SKIPPED 2008 too far back
region 13       -30,-60,-180,-150,2008  349.2857071213651       SKIPPED 2008 too far back
region 14       -30,-60,-150,-120,2008  369.6801779687019       SKIPPED 2008 too far back
region 15       -30,-60,-120,-90,2008   371.3506259204717       SKIPPED 2008 too far back
region 16       -30,-60,-90,-60,2010    372.34702161605793
region 17       -30,-60,-60,-30,2010    359.66173283160884
region 18       -30,-60,-30,0,2010      370.0846816336878
region 19       -30,-60,0,30,2010       371.34994353916517
region 20       -30,-60,30,60,2010      363.7914325581406
region 21       -30,-60,60,90,2008      370.62418424451164      SKIPPED 2008 too far back
region 22       -30,-60,90,120,2007     364.0968579581487       SKIPPED 2007 too far back
region 23       -30,-60,120,150,2007    376.6387005649721       SKIPPED 2007 too far back
region 24       -30,-60,150,180,2008    361.6299791449419       SKIPPED 2008 too far back
region 25       0,-30,-180,-150,2010    380.2447457627114
region 26       0,-30,-150,-120,2010    388.9062180066250 7
region 27       0,-30,-120,-90,2010     448.4514300572025
region 28       0,-30,-90,-60,2009      431.4154210526322       SKIPPED 2009 too far back
region 29       0,-30,-60,-30,2008      394.82310924369784      SKIPPED 2008 too far back
region 30       0,-30,-30,0,2009        404.39057971014495      SKIPPED 2009 too far back
region 31       0,-30,0,30,2000 360.7013833746901       SKIPPED 2000 too far back
region 32       0,-30,30,60,2008        388.5782183908046       SKIPPED 2008 too far back
region 33       0,-30,60,90,2002        368.9202919708033       SKIPPED 2002 too far back
region 34       0,-30,90,120,2007       373.9302706621559       SKIPPED 2007 too far back
region 35       0,-30,120,150,2007      362.7105999999999       SKIPPED 2007 too far back
region 36       0,-30,150,180,2009      383.0949243697488       SKIPPED 2009 too far back
region 37       30,0,-180,-150,2010     377.9182566341953
region 38       30,0,-150,-120,2010     372.3499269219533
region 39       30,0,-120,-90,2010      386.7785800240675
region 40       30,0,-90,-60,2010       395.4196434812996
region 41       30,0,-60,-30,2010       400.89539682539663
region 42       30,0,-30,0,2009 400.8142589928063       SKIPPED 2009 too far back
region 43       30,0,0,30,2010  372.15
```

```
region 44       30,0,30,60,1995 434.4135230141775       SKIPPED 1995 too far back
region 45       30,0,60,90,2007 389.78634212920775      SKIPPED 2007 too far back
region 46       30,0,90,120,2007        386.80046228710466      SKIPPED 2007 too far back
region 47       30,0,120,150,2007       341.7850960118169       SKIPPED 2007 too far back
region 48       30,0,150,180,2010       410.7476797552276
region 49       60,30,-180,-150,2010    291.42827633692883
region 50       60,30,-150,-120,2010    315.91298035866515
region 51       60,30,-120,-90,2010     376.08373983739824
region 52       60,30,-90,-60,2009      351.8672600475944       SKIPPED 2009 too far back
region 53       60,30,-60,-30,2009      374.4051229508205       SKIPPED 2009 too far back
region 54       60,30,-30,0,2008        365.64387239637307      SKIPPED 2008 too far back
region 55       60,30,0,30,2008 502.5995369274075       SKIPPED 2008 too far back
region 56       60,30,30,60,2008        454.8887256211605       SKIPPED 2008 too far back
region 57       60,30,120,150,2010      390.1156992687385
region 58       60,30,150,180,2010      395.7307636939459
region 59       90,60,-180,-150,2003    257.48198327359654      SKIPPED 2003 too far back
region 60       90,60,-150,-120,2003    289.67251414713064      SKIPPED 2003 too far back
region 61       90,60,-60,-30,2007      256.4065901183019       SKIPPED 2007 too far back
region 62       90,60,-30,0,2007        333.5862003977634       SKIPPED 2007 too far back
region 63       90,60,0,30,2008 499.8549204287123       SKIPPED 2008 too far back
rank 1  0,-30,-120,-90,2010,448.45143005720251494
rank 2  30,0,150,180,2010,410.74767975522757978
rank 3  -60,-90,-60,-30,2010,401.09499426511843012
rank 4  30,0,-60,-30,2010,400.89539682539663091
rank 5  60,30,150,180,2010,395.73076369394590301
rank 6  30,0,-90,-60,2010,395.41964348129960171
rank 7  60,30,120,150,2010,390.11569926873852409
rank 8  0,-30,-150,-120,2010,388.90621800662506757
rank 9  30,0,-120,-90,2010,386.77858002406748028
rank 10 0,-30,-180,-150,2010,380.24474576271137493
rank 11 30,0,-180,-150,2010,377.91825663419530201
rank 12 60,30,-120,-90,2010,376.08373983739824097
rank 13 30,0,-150,-120,2010,372.34992692195328346
rank 14 -30,-60,-90,-60,2010,372.34702161605792981
rank 15 30,0,0,30,2010,372.14999999999997726
```

The verbose option only affects the console output, not the output file contents.

Note that only 63 regions of 30x30 degrees are listed because some regions of the earth did not have any CO2 data readings in any year (e.g., non-ocean regions).

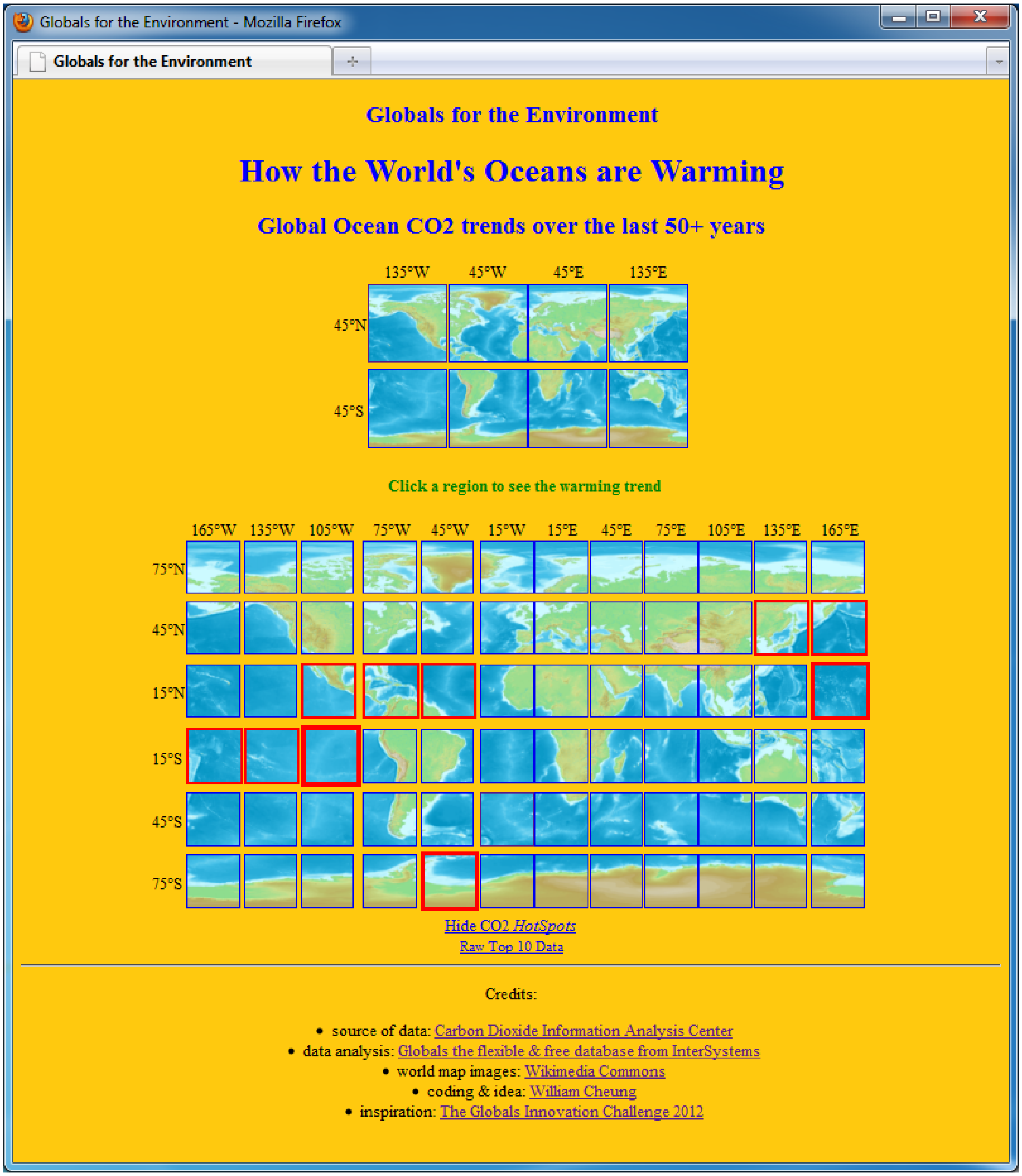
Also note that some average values were skipped because they were “too far back” from the latest data year. To control what years are skipped, use the -l and -m options (or -- latestYearWithData and --maxYearsToGoBackFromLatest). It turns out that the default values applied to the current LDEO dataset produces a top 10 rankings list from 2010 (averages from 2011 didn't make the top 10).

Visualizing the Heat Map using Top10OceanCO2Highlighter

As an enhancement to the *Globals for the Environment* web app, the output from the *GGSMR Hotspot Finder* was used to build a version of the home page where red outlining was applied to indicate the top 10 hotspot regions on the map. I wrote a custom Groovy script *Top10OceanCO2Highlighter* that reads both the hotspots CSV file and the homepage file and figures out which images linked in the homepage need the red outlining. The Groovy script applies the red outlining (image bordering) and outputs a new hotspots.html file with the red outlines, with a higher *intensity* for a higher CO2 level. The script is designed to be reusable with future changing top 10 lists as the LDEO researchers provide annual updates to the ocean CO2 dataset. Put the generated hotspots.html in the *public* directory of the web app, and run the app as described in the previous challenge:

node app.js

This is the result when you click the new link “Show top ten Ocean CO2 HotSpots” on the app's home page:



Try it yourself in the cloud: http://globals-for-the-environment.herokuapp.com

Conclusion

The flexibility and feature set of the Globals DB allowed me to quickly build a toolkit for big geospatial data analysis, valuable not only for environmental research but *any* research involving geospatial data. Using the complementary tools *GGSMR* and *GGSMR Hotspot Finder*, both built on Globals, does not require expensive infrastructure and developer resources. Simply install free Globals, Java, and Groovy on your workstation and go discover.