Towards a Domain-Specific Language for Geospatial Data Visualization Maps with Big Data Sets

Cleverson Ledur, Dalvan Griebler, Isabel Manssour, Luiz Gustavo Fernandes

Pontifical Catholic University of Rio Grande do Sul - PUCRS
Computer Science Graduate Program - PPGCC
Grupo de Modelagem de Aplicações Paralelas - GMAP

November 2015
Data Visualization

- Data visualization is the representation of data using graphic elements.
- Provide a quick understanding of data.
- Visualization creation workflow has three steps: data transformation, visual mapping and view transformation.

Figure: Visualization Creation Workflow
Users with low-level knowledge in programming that need to create a geo-spatial data visualization may have a hard time.

They will need to know at least JavaScript and HTML to create a simple map.

If they are dealing with a huge volume of data, it will be more difficult since most libraries and tools do not provide big data preprocessing.
Introduction

We propose an external *domain-specific language* for the creation of large-scale data visualization focused on web data visualization map applications. The main goal is to provide a *description language* that supports the visualization of the detail specification and the manipulation of raw data automatically, using a *data pre-processor*.

### Related Work

<table>
<thead>
<tr>
<th>DSL</th>
<th>Domain</th>
<th>Focus</th>
<th>Parallelism</th>
<th>Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViSlang</td>
<td>Sci/Vol Vis.</td>
<td>Vol Rend.</td>
<td>CPU/GPU</td>
<td>High-level(H:C++)</td>
</tr>
<tr>
<td>Diderot</td>
<td>Img Analysis &amp; Med. Vis.</td>
<td>Img Rend. &amp; Analysis</td>
<td>CPU/GPU</td>
<td>High-level(H:C)</td>
</tr>
<tr>
<td>Shadie</td>
<td>Med Vis.</td>
<td>Vol Rend.</td>
<td>CPU/GPU</td>
<td>High-level(H:Phyton)</td>
</tr>
<tr>
<td>Superconductor</td>
<td>General Interactive Vis.</td>
<td>Rend.</td>
<td>CPU/GPU</td>
<td>High-level(Ext.)</td>
</tr>
<tr>
<td>GMaVis</td>
<td>Geo-visualization</td>
<td>Data Proc.</td>
<td>CPU</td>
<td>Description Lan.(Ext.)</td>
</tr>
</tbody>
</table>

- **Vivaldi**, **ViSlang**, **Diderot**, and **Shadie** are focusing in the generation of volumetric data visualizations. **Superconductor** allows the user to create maps because have more expressiveness, but it requires programming skills.

- **Google Maps API**, **Leaflet** and **OpenLayers** are visualization maps libraries with high-level abstractions. However, they require users to learn a programming language and pre-processing, insert data.
Proposed DSL

- Aim at **facilitating the creation** of visualizations.
- To be as close as possible to the domain vocabulary (supporting a suitable and friendly language syntax).
- **Users will not have to know programming** aspects like functions, variables, methods and any other web development issue.
- User will have a **automatic data processing** that enables the data filtering, cleaning and classification.
- An **optimized file loading in memory** that allows open files bigger than the RAM memory available in the system.
Architecture Overview

**Figure:** DSL Environment

**Figure:** Data Pre-processor Workflow
Interface - Elements

- Description Language
- Few lines of code
- External DSL
- Manipulation and preprocessing of data

This DSL language rule set consists of blocks and declarations.

**Figure:** Example of Block and Declaration
Interface - Logical Operators

**Figure:** Logical Operators for Filtering and Classifying
Interface Example - Traffic Accidents in Porto Alegre (Brazil)

visualization: heatmap;

settings {
    latitude: field 42;
    longitude: field 41;
    page-title: "Acidentes Poa 2013";
    size: full;
    zoom-level: 11;
}

data {
    h-file: "acidentes-2013.csv";
    structure {
        delimiter: '\n';
        end-register: newline;
    }
}

**Figure:** Traffic accidents in Porto Alegre.
Interface Example - Flickr Pictures Classified by Brand of Used Camera

```plaintext
visualization: markedmap;
settings {
    latitude: field 12;
    longitude: field 11;
    marker-text: "<img src=" field 15 " width=200>>
    page-title: "Photos of 2014 by Camera";
    size: full;
}
data {
    file: "BIGDATA_YAHOO/yfcc100m_dataset-0";
    structure {
        delimiter: tab;
        end-register: newline;
        date-format: "YYYY-MM-DD";
    }
    filter: field 4 is greater than date "2014-02-01";
    classification {
        class ("Canon"): field 6 contains "Canon";
        class ("Sony"): field 6 contains "Sony";
        class ("Nikon"): field 6 contains "Nikon";
        class ("Panasonic"): field 6 contains "Panasonic";
        class ("Apple"): field 6 contains "Apple''
        class ("Kodak"): field 6 contains "Kodak";
    }
}
```

**Figure:** Flickr pictures by brand of used camera.
Evaluation

- **SLOCCount for code productivity.**
  - SLOCCount is a software measurement tool, which counts the physical source lines of code (SLOC), estimates development time, cost and effort based on the COCOMO model. It was used in [3, 5, 2, 1] researches.
  - Google Maps API, OpenLayers, and Leaflet comparison with the proposed DSL.
  - Four data visualization map applications.

- **Execution time for performance.**
  - Data processing with 10, 50 and 100 GB.
Data Set

- We used YFCC100M [4] data set as input.
- Yahoo Labs (https://labs.yahoo.com/).
- 54GB of data divided in 10 files delimited by a tab character.
- This is a public multimedia data set with 99.3 million images and 0.7 million videos from Flickr.
- ID, Tags, Urls, date, time, device...
Figure: SLOCCount Programming Code productivity Results

Table: Code productivity (physical source lines of code)

<table>
<thead>
<tr>
<th>Application</th>
<th>DSL</th>
<th>Google Maps API</th>
<th>OpenLayers</th>
<th>Leaflet</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Devices”</td>
<td>22</td>
<td>74</td>
<td>25</td>
<td>79</td>
</tr>
<tr>
<td>“War”</td>
<td>15</td>
<td>20</td>
<td>27</td>
<td>25</td>
</tr>
<tr>
<td>“Manifestation”</td>
<td>15</td>
<td>20</td>
<td>17</td>
<td>25</td>
</tr>
<tr>
<td>“World Cup”</td>
<td>15</td>
<td>20</td>
<td>18</td>
<td>25</td>
</tr>
</tbody>
</table>
## Completion Times

**Table:** Completion Times (Seconds)

<table>
<thead>
<tr>
<th>Size</th>
<th>Data Transformation</th>
<th>Visual Mapping - Google Maps API</th>
</tr>
</thead>
<tbody>
<tr>
<td>10GB</td>
<td>110.4948 (Std. 0.9763)</td>
<td>2.910 (Std. 1.6084)</td>
</tr>
<tr>
<td>50GB</td>
<td>544.0506 (Std. 9.4225)</td>
<td>3.2738 (Std. 2.0663)</td>
</tr>
<tr>
<td>100GB</td>
<td>1098.9284 (Std. 19.0383)</td>
<td>3.8536 (Std. 2.7584)</td>
</tr>
</tbody>
</table>
Final Remarks

- We provided a new domain-specific language for the creation of geospatial data visualization with simpler and friendly interface.
- Our DSL may help data visualization users for gaining insights and extracting information from big data sets.
- Evaluation demonstrates that it increases the user’s productivity by the possibility of automatically handling raw input data.
Final Remarks

We plan as future work:

- To investigate alternatives for **taking advantage of the parallelism** available on the multi-core architectures, speeding the pre-processing performance.
- To provide **more features in our language interface**. For example, new visualization types (graphs, treemaps, column and area charts), and new logical operators for data selection.
- To **support input of JSON and XML**.
- To include some **advanced classification with data mining** algorithms in our data processor.
References I

A Unified MapReduce Domain-Specific Language for Distributed and Shared Memory Architectures.

Performance and Usability Evaluation of a Pattern-Oriented Parallel Programming Interface for Multi-Core Architectures.

M. Hertz, Y. Feng, and E. D. Berger.
Garbage Collection Without Paging.

The New Data and New Challenges in Multimedia Research.

Refinement Types for Haskell.
Thank you!

Questions & Answers

cleversonledur@gmail.com