GMaVis: A Domain-Specific Language for Large-Scale Geospatial Data Visualization Supporting Multi-core Parallelism

Cleverson Ledur
Advisor: Ph.D. Luiz Gustavo Fernandes
Co-Advisor: Ph.D. Isabel Manssour

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

November 2015

Outline

1. Introduction
2. Related Work
3. GMaVis: A DSL for Geospatial Data Visualization
4. GMaVis' Compiler
5. Evaluation and Discussion
6. Conclusion

1 **Introduction**
- Contextualization
- GMaVis Introduction

2 Related Work

3 GMaVis: A DSL for Geospatial Data Visualization

4 GMaVis' Compiler

5 Evaluation and Discussion

6 Conclusion

Data Visualization

- Data visualization is the representation of data using graphic elements.
- Provide a quick understanding of data.
- Visualization creation pipeline has three steps: data pre-processing, data to visual mappings, and view transformations.

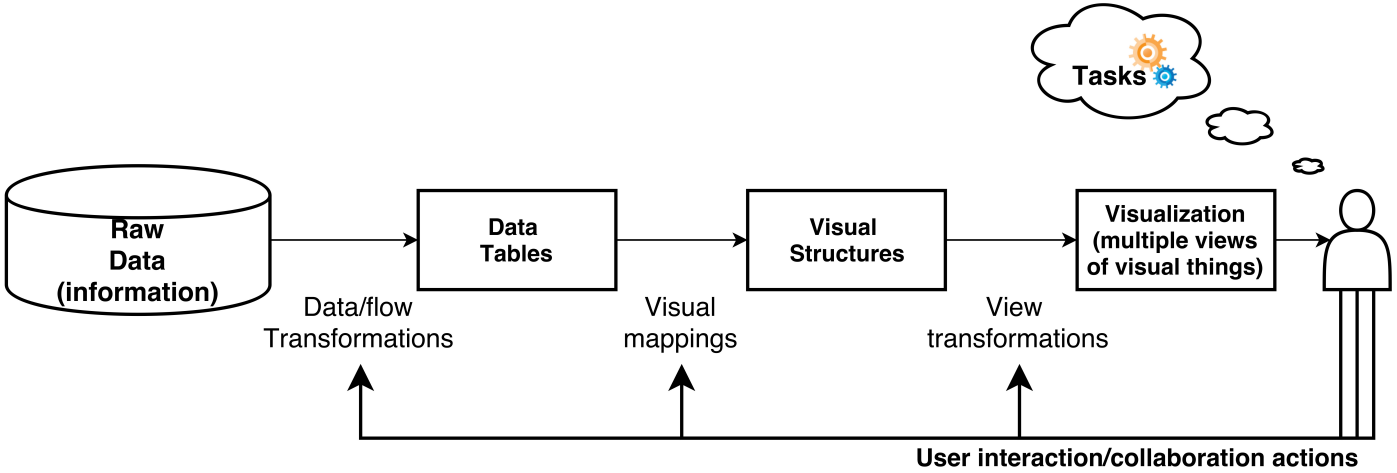


Figure: Visualization pipeline (Adapted from [1])

Data Visualization

- Users with low-level knowledge in programming that need to create a geo-spatial data visualization may have a hard time.
- For example, to create a simple web visualization map, they will need to know at least JavaScript and HTML.
- 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.
- It is possible to use parallel processing to speed up visualization creation.

Parallel Programming

- Developers must worry about:
  - Architecture System?
  - Parallel Interface?
  - Strategy: Shared Memory, Message Passing, Hybrid.
  - Decomposition of the problem: Data Parallelism, Task Parallelism, Stream Parallelism.
  - Steps: Study problem or code, Look for parallelism opportunities, Try to keep all cores busy.
  - Developer must care about Synchronization (Locks/Semaphores, Synchronous through communication, Barriers), Data Dependencies, Granularity, I/O, ...

1 **Introduction**
- Contextualization
- GMaVis Introduction

2 **Related Work**

3 **GMaVis: A DSL for Geospatial Data Visualization**

4 **GMaVis' Compiler**

5 **Evaluation and Discussion**

6 **Conclusion**

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 **high-level interface** that supports the visualization of the detail specification and the **manipulation of raw data automatically**, using a **parallel data pre-processor**.

Introduction

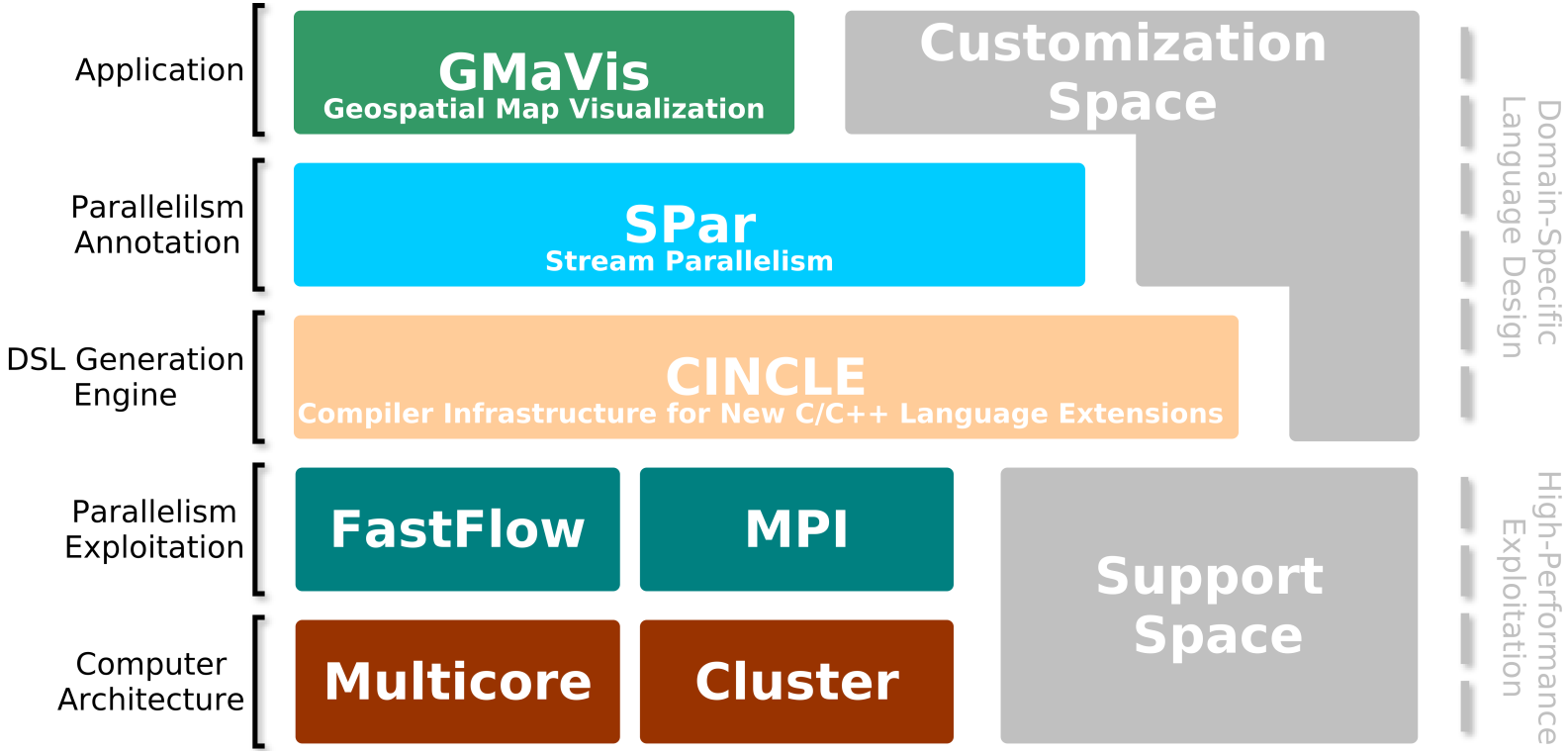


Figure: Research scenario framework (Extracted from [4]).

1 Introduction

2 Related Work
- DSLs for Visualization

3 GMaVis: A DSL for Geospatial Data Visualization

4 GMaVis' Compiler

5 Evaluation and Discussion

6 Conclusion

Related Work

DSL	Domain	Focus	Parallelism	Interface
Vivaldi	Sci/Vol Vis.	Vol Rend.	Dist. CPU/GPU	High-level (H:Phyton)
ViSlang	Sci/Vol Vis.	Vol Rend.	CPU/GPU	High-level(H:C++)
Diderot	Img Analysis & Med. Vis.	Img Rend. & Analysis	CPU/GPU	High-level(H:C)
Shadie	Med Vis.	Vol Rend.	CPU/GPU	High-level(H:Phyton)
Superconductor	General Interactive Vis.	Rend.	CPU/GPU	High-level(Ext.)
GMaVis [6]	Geo. Data Vis. Maps	Data Preprocessing	CPU	High-level(Ext.)

- **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 do not avoid to learn a programming language, pre-processes and insert data.

Related Work

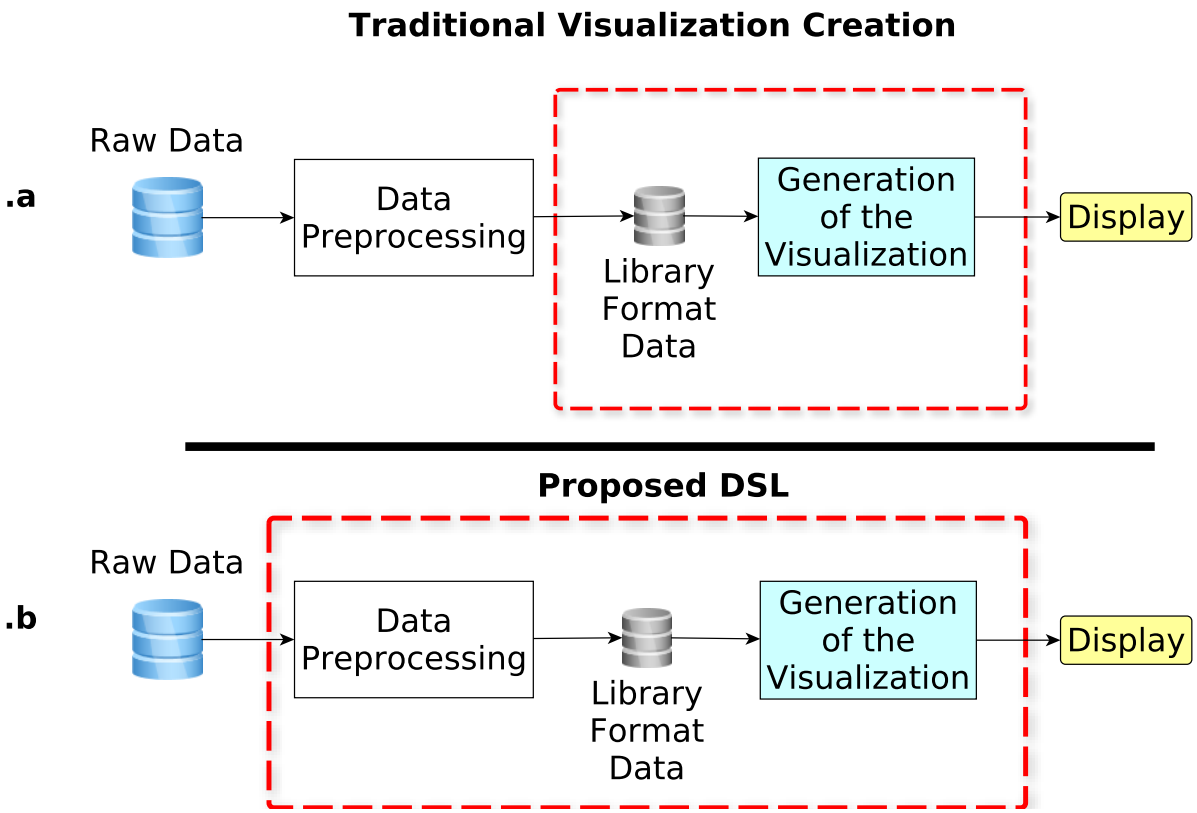


Figure: Data Visualization Creation Comparison

Related Work

Table: Complexities abstraction in each visualization creation phase.

DSL	Data Pre-processing	Data to Visual Mappings	View Transformations
Vivaldi [3]		X	X
ViSlang [8]		X	X
Diderot [2]		X	X
Shadie [5]		X	X
Superconductor [7]		X	X
GMaVis [6]	X	X	X

Table: Parallel processing in each visualization creation phase.

DSL	Data Pre-processing	Data to Visual Mappings	View Transformations
Vivaldi [3]		X	X
ViSlang [8]		X	X
Diderot [2]		X	X
Shadie [5]		X	X
Superconductor [7]		X	X
GMaVis [6]	X		

1 Introduction

2 Related Work

3 **GMaVis: A DSL for Geospatial Data Visualization**
- GMaVis

4 GMaVis' Compiler

5 Evaluation and Discussion

6 Conclusion

Proposed DSL

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

Architecture Overview

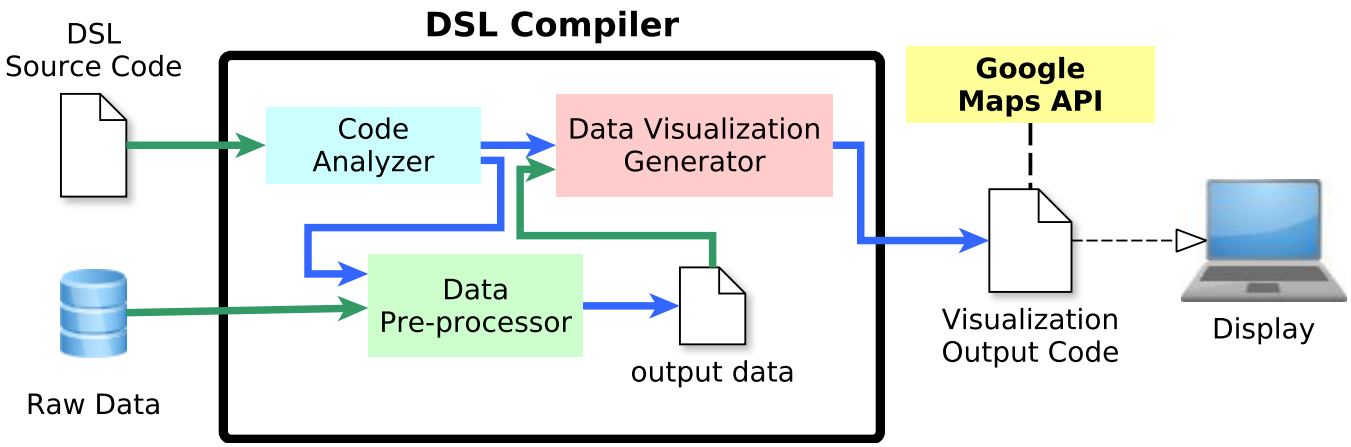


Figure: DSL Enviroment

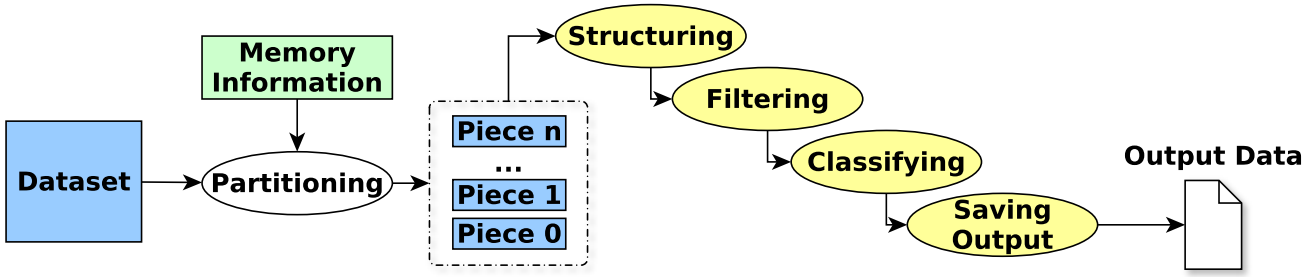


Figure: Data Pre-processor Workflow

Interface - Elements

- High-level interface
- Few lines of code
- External DSL
- Manipulation and preprocessing of data

This DSL language rule set consists of blocks and declarations.

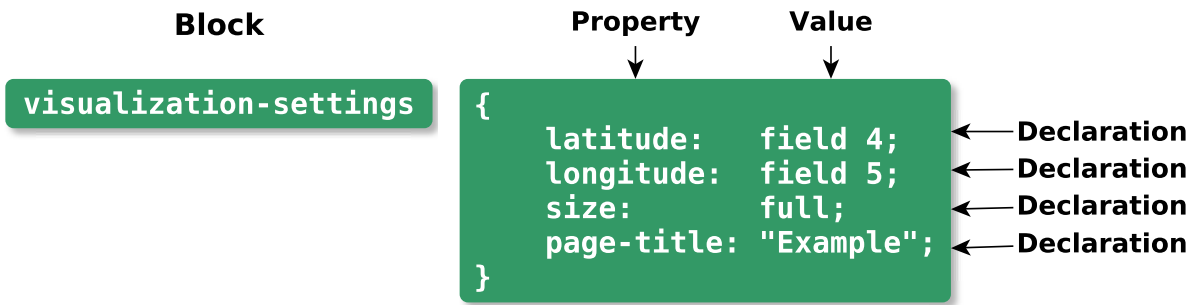


Figure: Example of Block and Declaration

Interface - Logical Operators

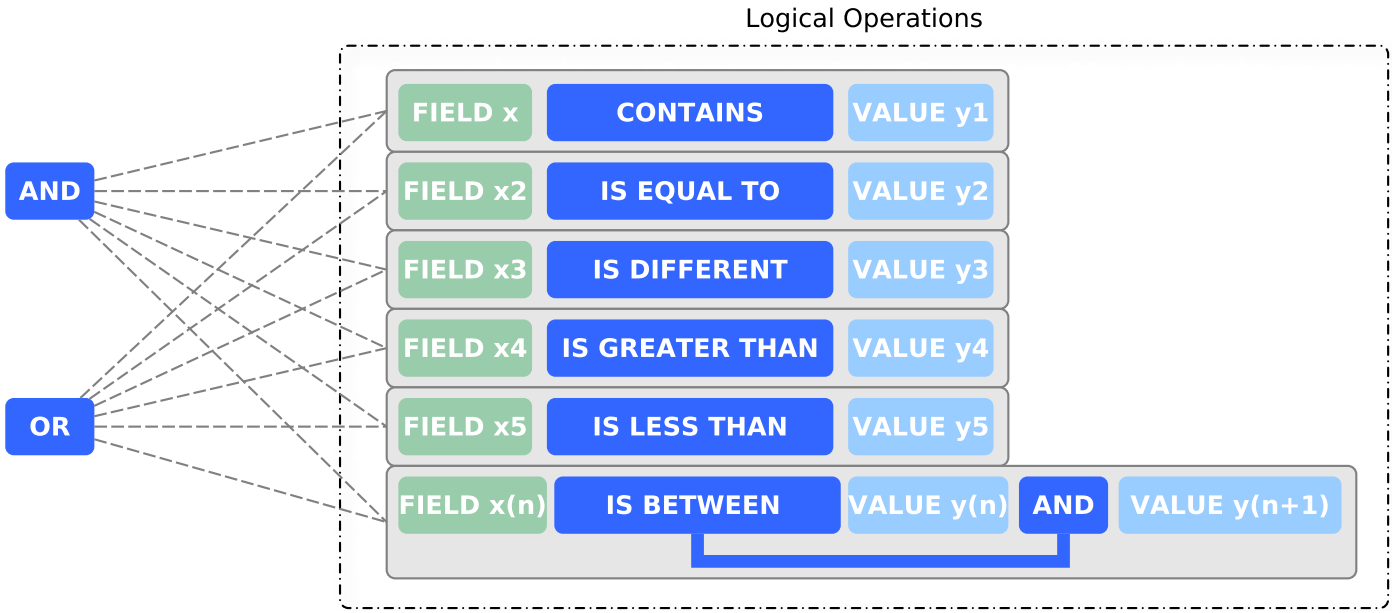


Figure: Logical Operators for Filtering and Classifying

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: ';';
            end-register: newline;
      }
}
```

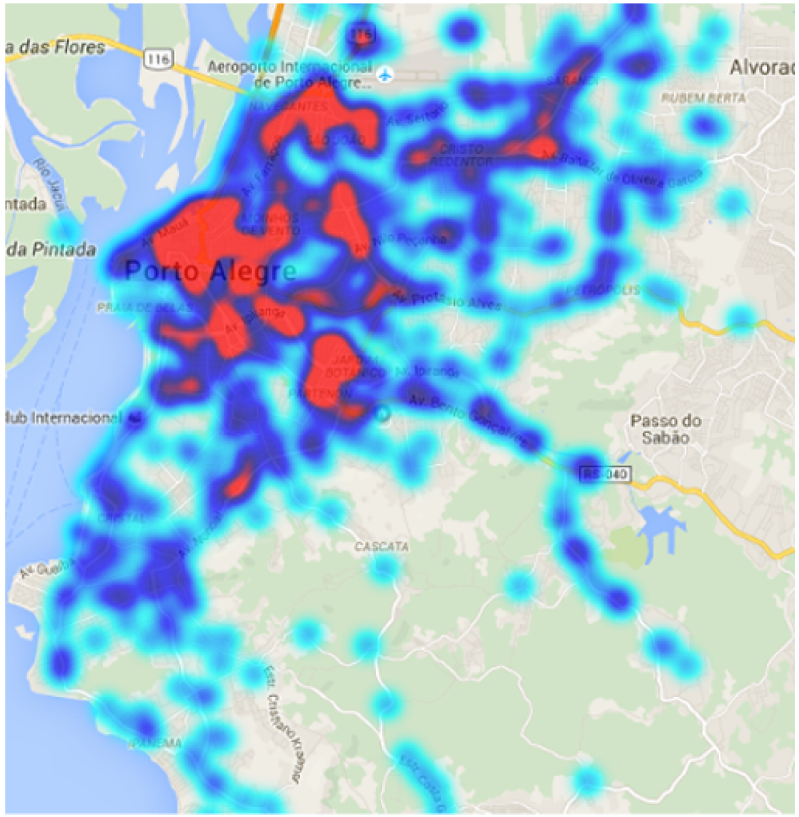


Figure: Traffic accidents in Porto Alegre.

Interface Example - Flickr Pictures Classified by Brand of Used Camera

```
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 camera used.

1 Introduction

2 Related Work

3 GMaVis: A DSL for Geospatial Data Visualization

4 GMaVis' Compiler
- Compiler Environment
- Data Preprocessor Generator
- Parallel Data Preprocessor

5 Evaluation and Discussion

6 Conclusion

Compiler Environment

- Flex and Bison
- Receives the source code and performs lexical and syntax analysis.
- Generates tokens and join them according to the specified grammar rules.
- Each rule has different actions, such as: saving information, calling functions to process values, concatenation of strings and flagging.

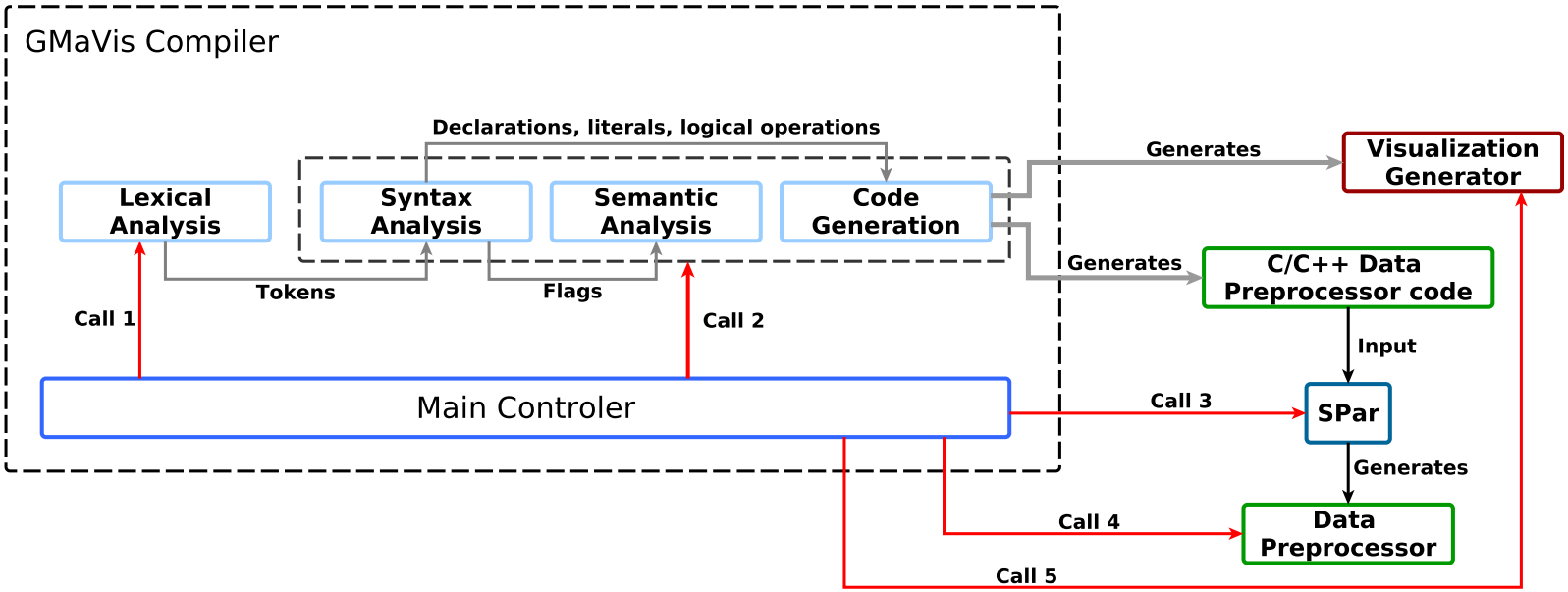


Figure: GMaVis compiler environment.

1 Introduction

2 Related Work

3 GMaVis: A DSL for Geospatial Data Visualization

4 GMaVis' Compiler
- Compiler Environment
- Data Preprocessor Generator
- Parallel Data Preprocessor

5 Evaluation and Discussion

6 Conclusion

Data Preprocessor Generator

Figure: Example of logical expression transformation.

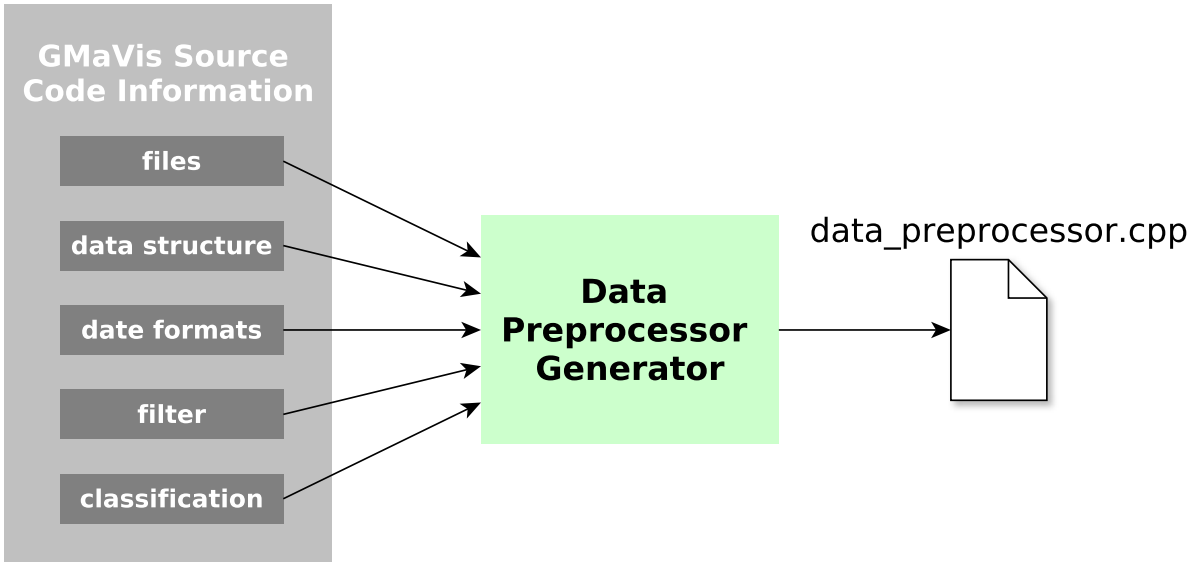


Figure: Data preprocessor generation.

Data Preprocessor Generator

Table: Data preprocessor functions for each logical operator.

GmaVis Logical Operator	Input Data Type	Generated Function in Data Preprocessor
Is equal to	string	string_is_equal(*counter,data,i,field,value*)
	date	date_is_equal(*counter,data,i,field,date2*)
	integer	int_is_equal(*counter,data,i,field,value*)
	float	float_is_equal(*counter,data,i,field,value*)
Is different than	string	string_is_different(*counter,data,i,field,value*)
	date	date_is_different(*counter,data,i,field,date2*)
	integer	int_is_different(*counter,data,i,field,value*)
	float	float_is_different(*counter,data,i,field,value*)
Is greater than	date	date_is_greater(*counter,data,i,field,date2*)
	integer	int_is_greater(*counter,data,i,field,value*)
	float	float_is_greater(*counter,data,i,field,value*)
Is less than	date	date_is_less(*counter,data,i,field,date2*)
	integer	int_is_less(*counter,data,i,field,value*)
	float	float_is_less(*counter,data,i,field,value*)
Is between and	date	date_is_between(*counter,data,i,field,date2,date3*)
	integer	int_is_between(*counter,data,i,field,value1,value2*)
	float	float_is_between(*counter,data,i,field,value1,value2*)
contains	string	string_contains(*counter,data,i,field,value*)

Visualization Generator

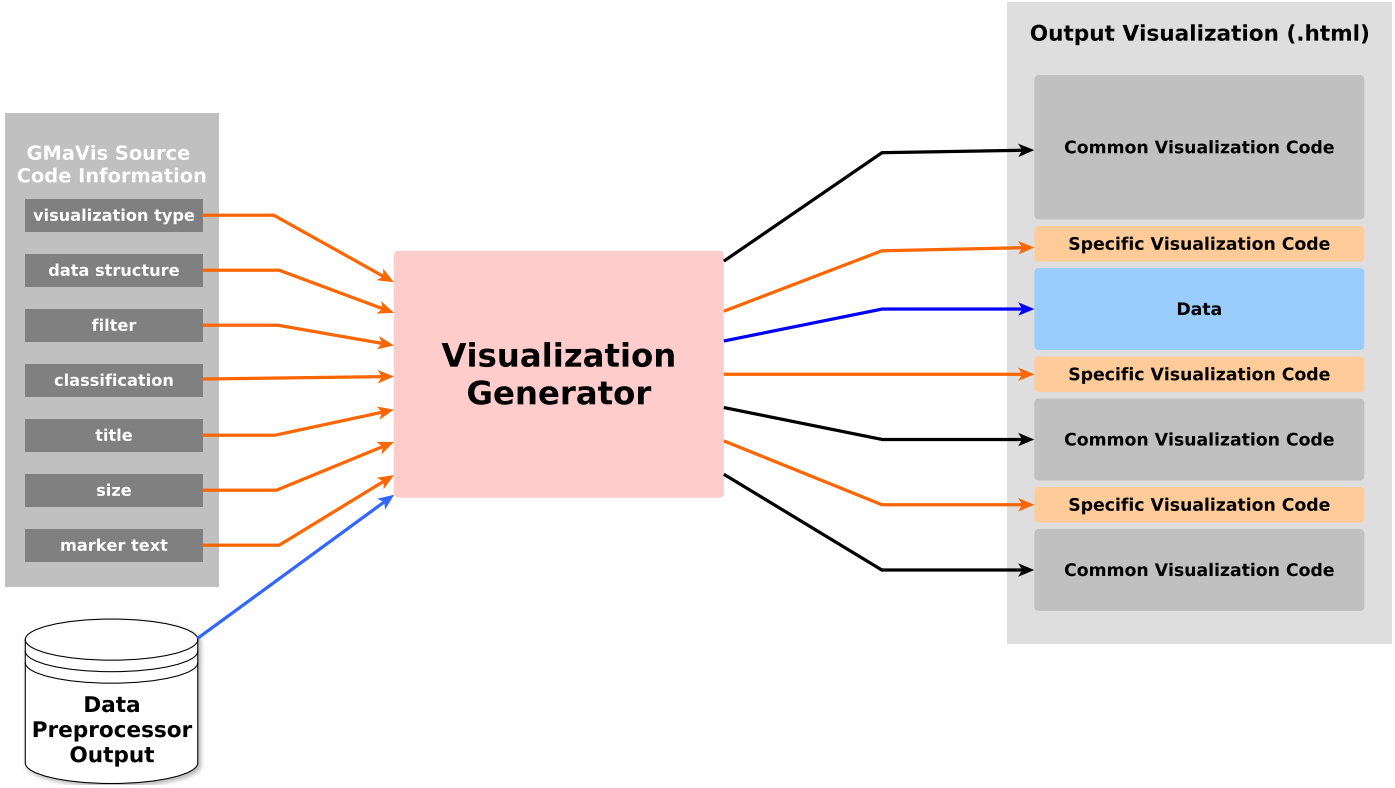


Figure: Data visualization generation.

1 Introduction

2 Related Work

3 GMaVis: A DSL for Geospatial Data Visualization

4 GMaVis' Compiler
- Compiler Environment
- Data Preprocessor Generator
- Parallel Data Preprocessor

5 Evaluation and Discussion

6 Conclusion

Parallel Data Preprocessor

- Data preprocessor module has a **high computational cost** [6].
- We implemented a parallel version for **multi-core** architectures.
- The code that we implement to parallelize the data preprocessor using **SPar** is almost the same as is used in the sequential version.
- GMaVis compiler **generates SPar annotations**.

Sequential Data Preprocessor

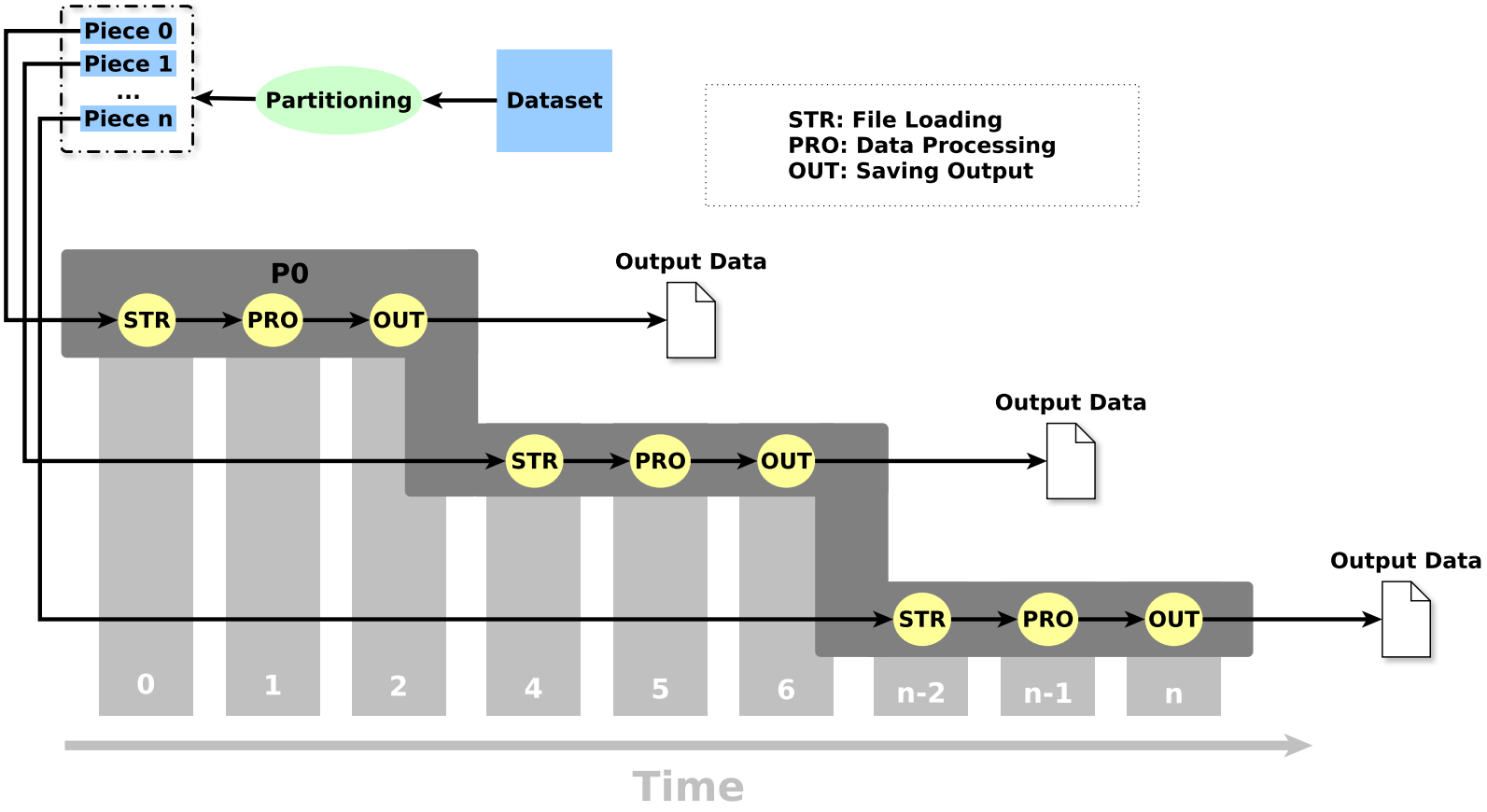


Figure: Example of sequential processing in data preprocessor.

Parallel Data Preprocessor

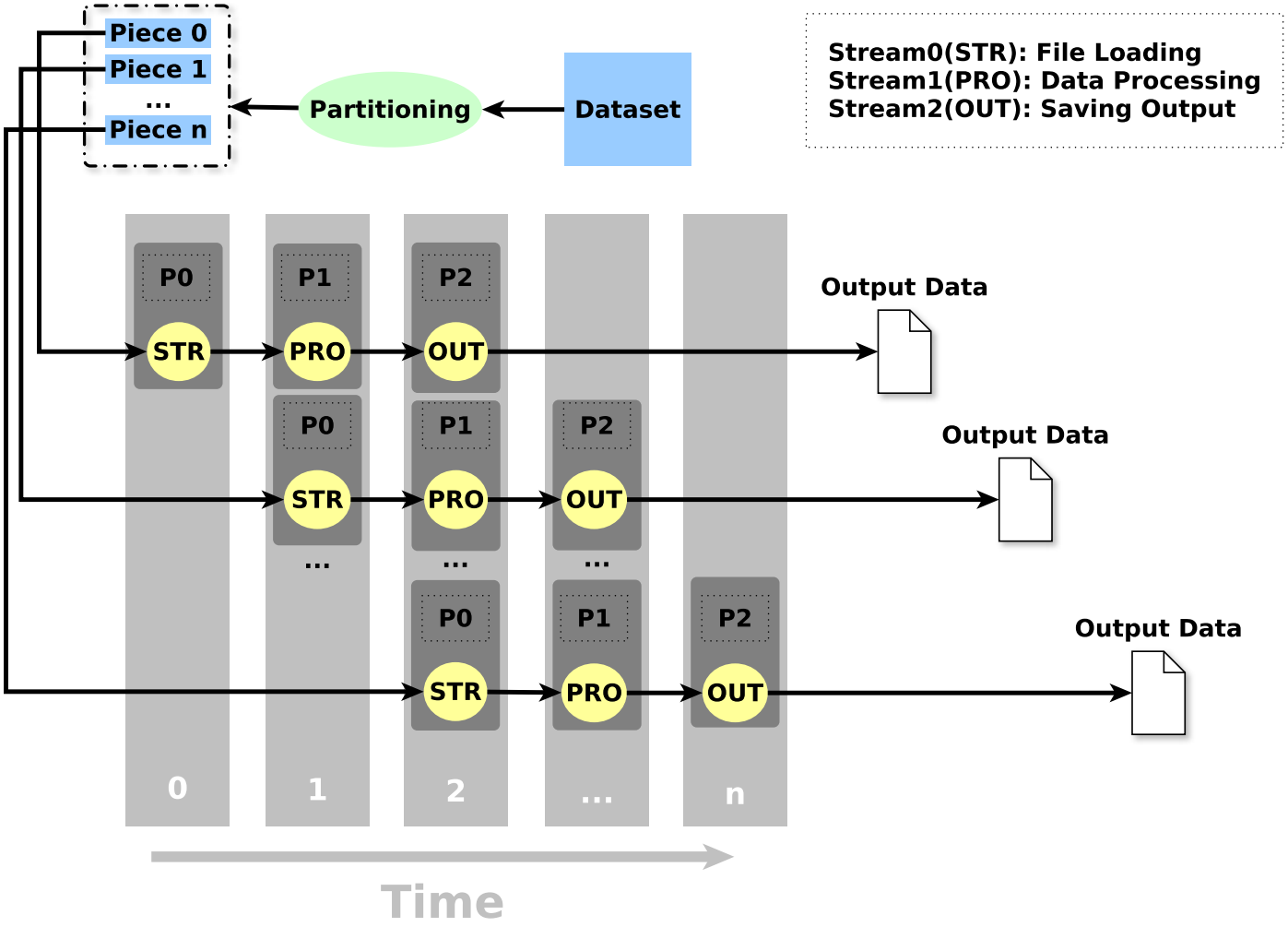


Figure: Parallel preprocessor using pipeline strategy.

Data Preprocessor Generator

```
void process(...)
{
        variables declaration
        calculate crunch size
        open file
        while(last_read < file_size)
        {
                read a chunk from the file.
                verify its consistency

                start processing chunk
                returns a string with the output

                writes the output string in the output file
                deallocates memory for chunk read
        }
        close file
}
```

Figure: Illustration of sequential process function code.

Data Preprocessor Generator

```
void process(...)
{
        variables declaration
        calculate crunch size
        open file
        [[spar::ToStream(),spar::Input(...)]]
        while(last_read < file_size)
        {
                read a chunk from the file.
                verify data consistency
                [[spar::Stage(),spar::Input(...),spar::Output(...),spar::Replicate(...)]]
                {
                        start processing chunk
                        returns a string with the output
                }
                [[spar::Stage(),spar::Input(...)]]
                {
                        writes the output string in the output file
                        deallocates memory for chunk read
                }
        }
        close file
}
```

Figure: Illustration of process function code with SPar annotations.

1 Introduction

2 Related Work

3 GMaVis: A DSL for Geospatial Data Visualization

4 GMaVis' Compiler

5 **Evaluation and Discussion**
- Methodology
- Programming Effort Results
- Performance Results

6 Conclusion

Research Questions

- **Q1 - Is it possible to create a DSL (GMAVIS) to reduce the programming effort for creating geospatial visualization?**

Research Questions

- **Q1 - Is it possible to create a DSL (GMAVIS) to reduce the programming effort for creating geospatial visualization?**
  - **H1**- GMaVis requires less programming effort than visualization libraries (Google Maps API, Leaflet and OpenLayers).

Research Questions

- **Q1 - Is it possible to create a DSL (GMAVIS) to reduce the programming effort for creating geospatial visualization?**
  - **H1**- GMaVis requires less programming effort than visualization libraries (Google Maps API, Leaflet and OpenLayers).
  - **H2**- The inclusion of automatic data pre-processing in GMaVis reduces programming effort.

Research Questions

- **Q1 - Is it possible to create a DSL (GMAVIS) to reduce the programming effort for creating geospatial visualization?**
  - **H1**- GMaVis requires less programming effort than visualization libraries (Google Maps API, Leaflet and OpenLayers).
  - **H2**- The inclusion of automatic data pre-processing in GMaVis reduces programming effort.
- **Q2 - Can the parallel code generated by this DSL speed up the data processing of raw geospatial data?**

Research Questions

- **Q1 - Is it possible to create a DSL (GMAVIS) to reduce the programming effort for creating geospatial visualization?**
  - **H1**- GMaVis requires less programming effort than visualization libraries (Google Maps API, Leaflet and OpenLayers).
  - **H2**- The inclusion of automatic data pre-processing in GMaVis reduces programming effort.
- **Q2 - Can the parallel code generated by this DSL speed up the data processing of raw geospatial data?**
  - **H3**- GMaVis can generate parallel code annotations using SPar for speeding up performance.

Research Questions

- **Q1 - Is it possible to create a DSL (GMAVIS) to reduce the programming effort for creating geospatial visualization?**
  - **H1**- GMaVis requires less programming effort than visualization libraries (Google Maps API, Leaflet and OpenLayers).
  - **H2**- The inclusion of automatic data pre-processing in GMaVis reduces programming effort.
- **Q2 - Can the parallel code generated by this DSL speed up the data processing of raw geospatial data?**
  - **H3**- GMaVis can generate parallel code annotations using SPar for speeding up performance.
  - **H4** - Code generation for SPar is simpler than using TBB (Thread Building Blocks) for speeding up performance.

Applications - Datasets

- **YFCC100M:** Provided by Yahoo Labs. It has about 54GB of data, divided into ten files. This is a public multimedia data set with 99.3 million images and 0.7 million videos, all from Flickr and under Creative Commons licensing.
- **Traffic Accidents:** Also we used a dataset from DataPoa with traffic accident data in Porto Alegre, Brazil that contains information about the type of accidents, vehicles, date and time, level and location. It has 39 fields, including latitude and longitude, and about 20.937 registers.
- **Airports:** A dataset obtained in OpenFlights[1] that contains all the airports in the world was used, with information about latitude and longitude, city, country and airport code. It has 8107 registers with 12 fields with information about the airports.

- Six visualization applications were created using these datasets.
- Two applications for each type of visualization provided in GMaVis (MarkedMap, Heatmap, and Clusteredmap).
- GMaVis, Google Maps API, Leaflet and OpenLayers.
- C/C++ data preprocessor were created for Google Maps API, Leaflet and OpenLayers.

Evaluation

- Q1 - Programming effort evaluation
  - COCOMO Model (Sloccount)
- Q2 - Performance evaluation
  - Execution Time
  - Throughput

Programming Effort Evaluation

- This Validates the effectiveness of facilitating visualization creation with GMaVis.
- This evaluation compares the effort to create a visualization using GMaVis and traditional tools.
- Traditional tools: Google Maps API, Leaflet and OpenLayers.
- 6 Applications (2 for each visualization type).
- In order to estimate data preprocessing, we measured a code developed in C++ that loads, processes data and output in the format required by the application.

SLOCCount

- COCOMO model
  - Usability model for measuring code and estimation metrics.
  - Development time and effort based on the physical source lines of code (SLOC/KSLOC).
  - **Entire development cycle** for generating a visualization, including the initial process of planning, coding, testing, documenting and deploying it for users.
- SLOCCount tool
  - It is a software measurement tool, which counts the physical source lines of code (SLOC), ignoring empty lines and comments.
  - It also estimates development time, cost and effort based on the original Basic COCOMO model.

Performance Evaluation

- It evaluates the parallel data preprocessor to verify if it achieves better performance compared to the sequential version.
- Since TBB was also considered to be used to parallelize data preprocessor module because it offers stream parallelism support, we compared its aplication to SPar.
- Verifies if a parallelized TBB version presents better results than the automatic generated SPar version of data preprocessor.

Performance Evaluation - Hardware/Software

Multi-core computer Blade Dell PowerEdge M610 with the following specification:

- Two processors Intel Xeon Six-Core E5645 2.4GHz Hyper-Threading
- 24 GBytes of memory.
- Software:
  - Operational System: Ubuntu 14.04 LTS
  - C/C++ Compiler: G++ 5.2
  - SPar Compiler: CINCLE

1 Introduction

2 Related Work

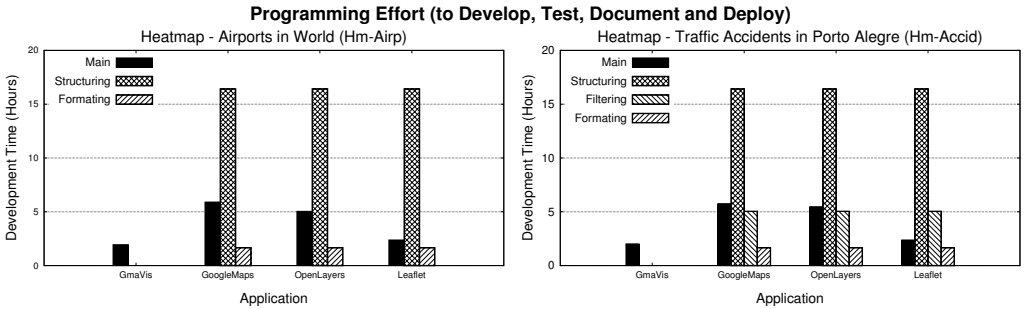
3 GMaVis: A DSL for Geospatial Data Visualization

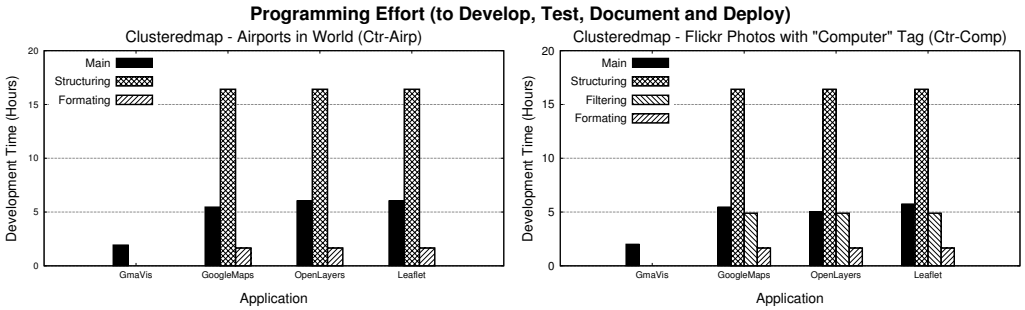
4 GMaVis' Compiler

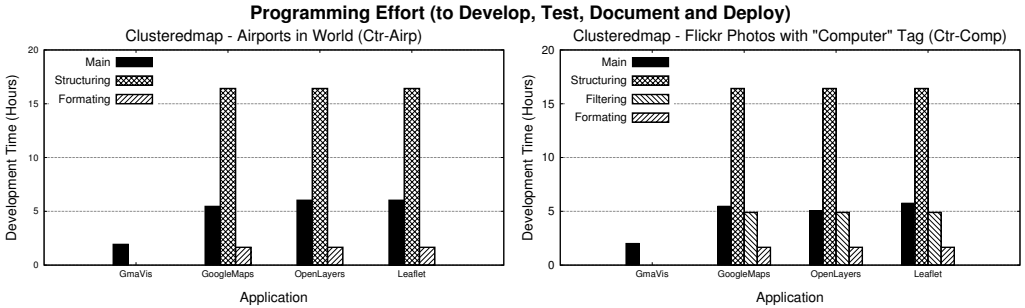
5 Evaluation and Discussion
- Methodology
- Programming Effort Results
- Performance Results

6 Conclusion

Effort Evaluation - Programming Effort (Entire development cycle).







Code productivity results

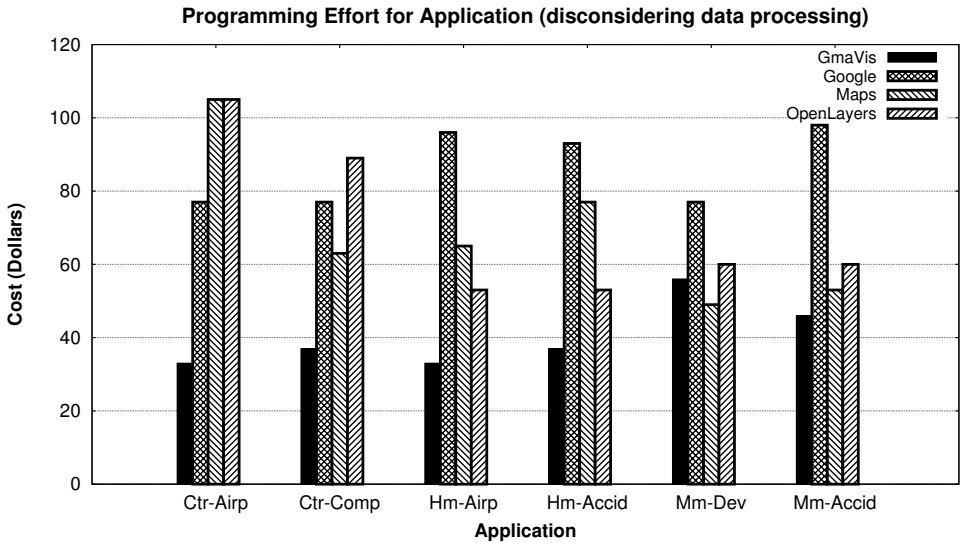


Table: SLOC (Physical Lines of Code) for each application.

Application	GmaVis	Google Maps	OpenLayers	Leaflet
Ctr-Airp	15	34	**46**	**46**
Ctr-Comp	17	34	28	39
Hm-Airp	**15**	42	29	24
Hm-Accid	17	41	34	24
Mm-Dev	25	34	22	27
Mm-Accid	21	43	24	27

Lines of code to parallelize data preprocessor using SPar and TBB

Application	Sequential	SPAR		TBB	
	SLOC	SLOC	***Code Increase*** (%)	SLOC	***Code Increase*** (%)
CM_CP	210	216	2,857	274	30,476
CM_AIR	199	205	3,015	263	**32,160**
HM_AC	210	216	2,857	274	30,476
HM_AIR	199	205	3,015	263	**32,160**
MM_DEV	224	230	**2,678**	288	28,571
MM_ACID	216	222	2,777	280	29,629

1 Introduction

2 Related Work

3 GMaVis: A DSL for Geospatial Data Visualization

4 GMaVis' Compiler

5 Evaluation and Discussion
- Methodology
- Programming Effort Results
- Performance Results

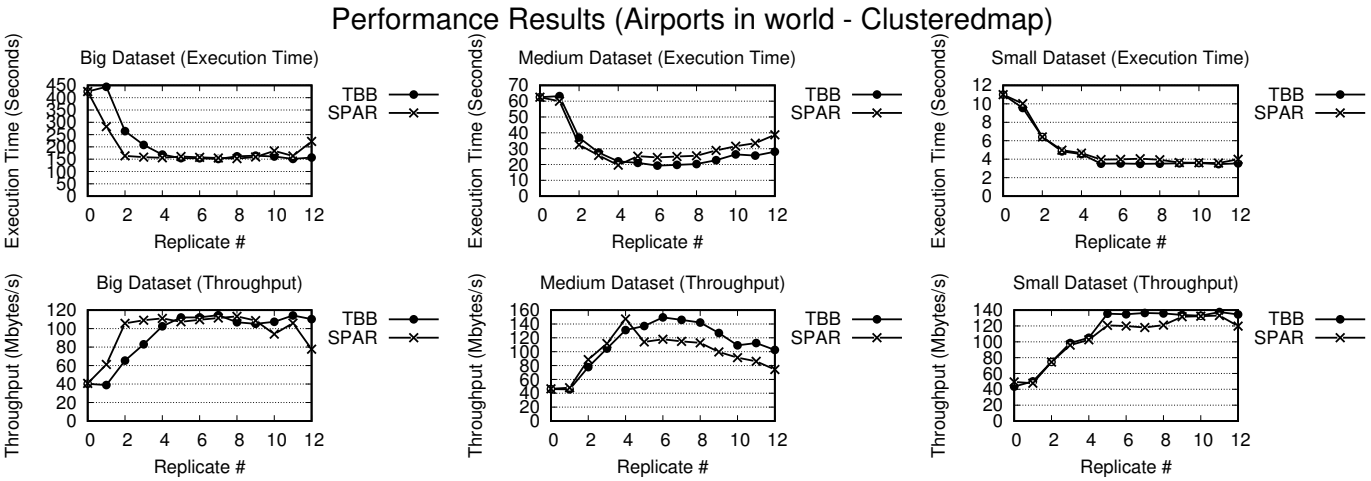
6 Conclusion

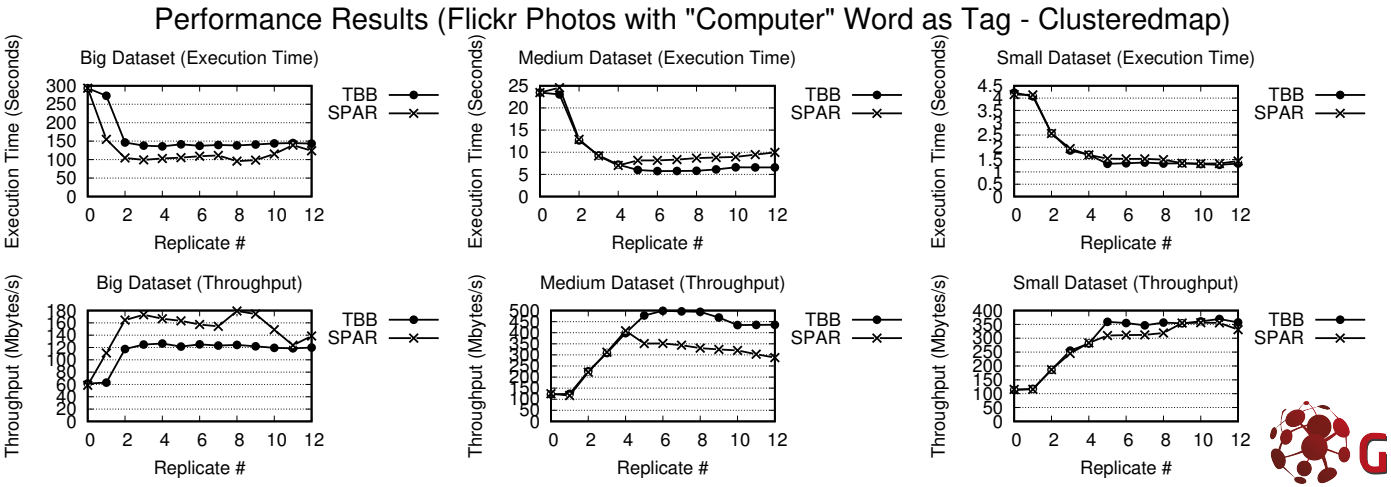
Sequential GMaVis Execution Times

Table: Completion times (seconds) [6].

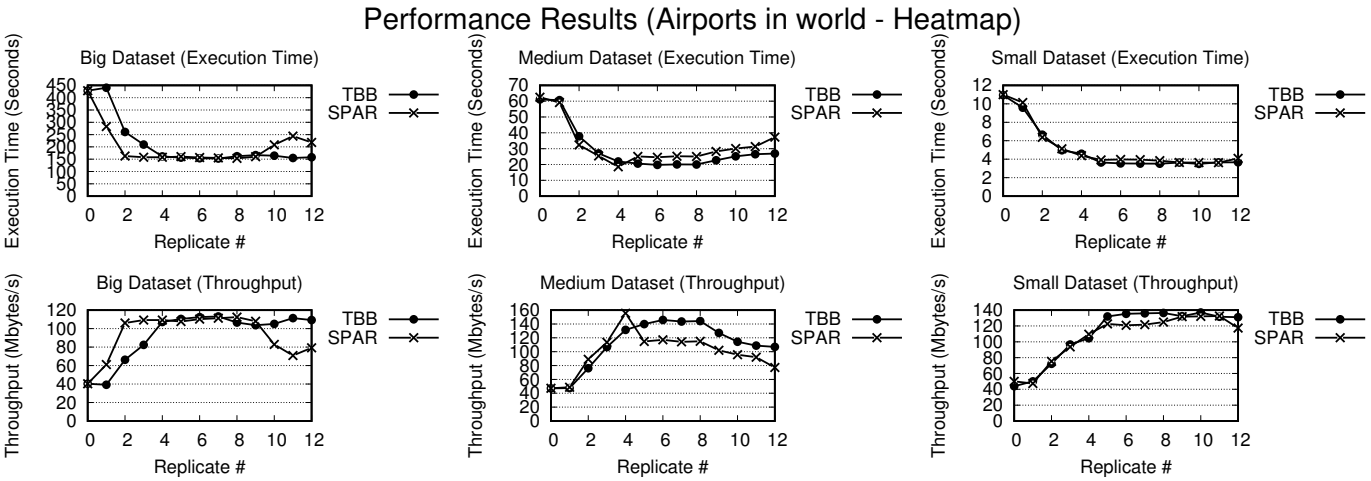
Size	Data Pre-processing	Data to Visual Mappings - Google Maps API
10GB	110.4948 (Std. 0.9763)	2.910 (Std. 1.6084)
50GB	544.0506 (Std. 9.4225)	3.2738 (Std. 2.0663)
100GB	1098.9284 (Std. 19.0383)	3.8536 (Std. 2.7584)

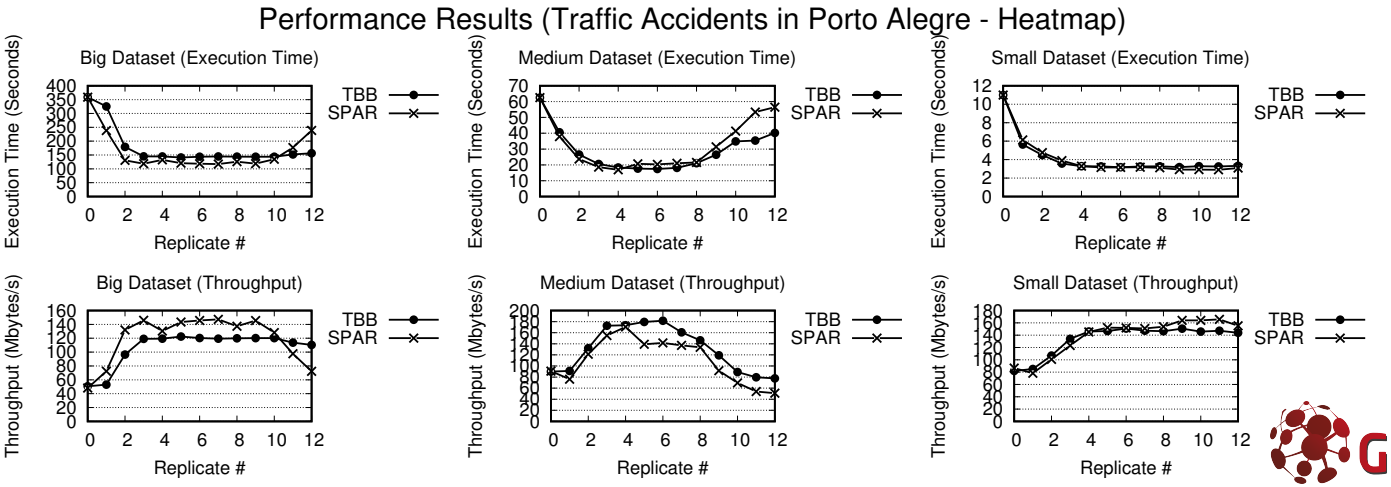
TBB and SPar SLOC



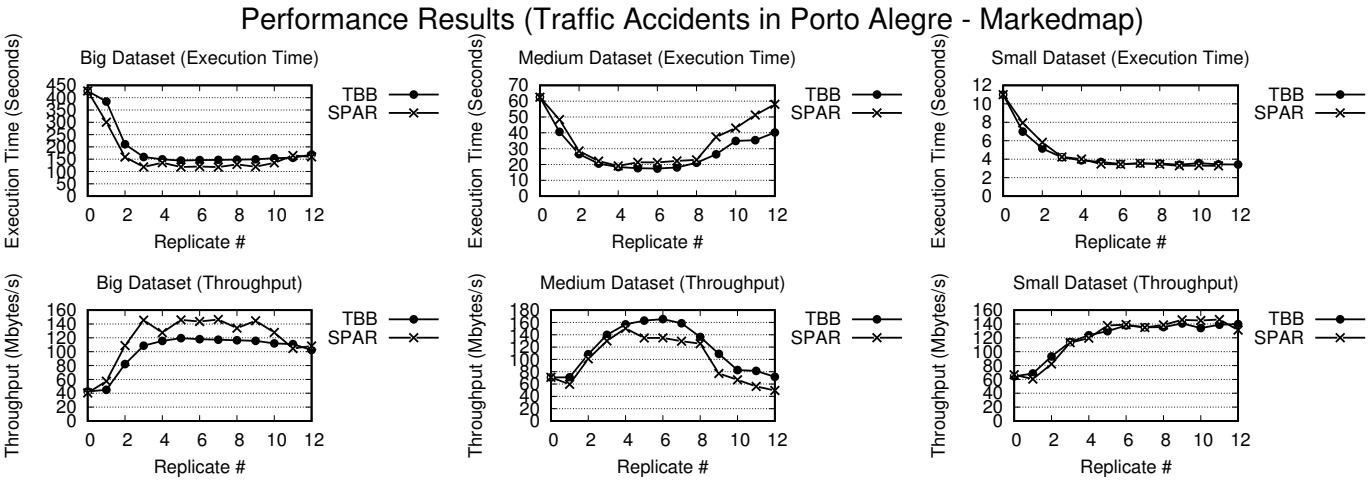


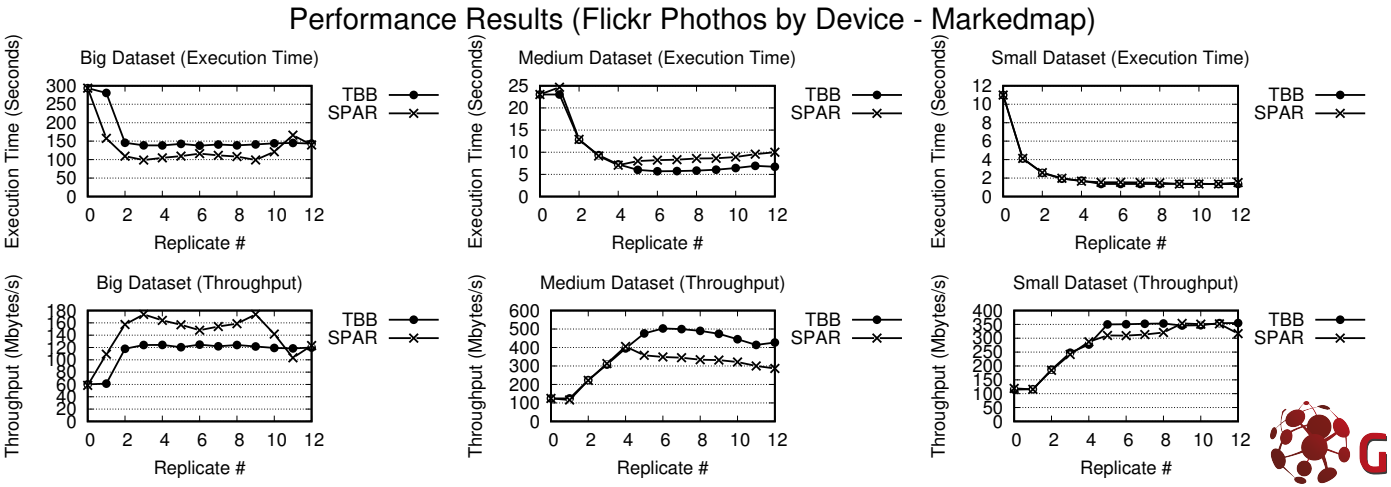
TBB and SPar SLOC





TBB and SPar SLOC





1 Introduction

2 Related Work

3 GMaVis: A DSL for Geospatial Data Visualization

4 GMaVis' Compiler

5 Evaluation and Discussion

6 Conclusion
- Conclusion

Final Remarks

- We provided a new domain-specific language for the geospatial data with simpler and friendly interface.
- It offers support for big raw data sets.
- We used a real-world data set to evaluate our DSL, comparing it in different visualization types.

Final Remarks - Research Questions

Q1 - *Is it possible to create a DSL (GMAVIS) to reduce the programming effort for creating geospatial visualization?*

- Results demonstrate that even when providing the abstraction of complexities in the *data pre-processing* phase and totally parallel programming abstraction, GMaVis could reduce the effort and cost required to implement the three supported data visualizations (clusteredmap, heatmap, and markedmap) **(Validades H1)**.
- It is possible to confirm that GMaVis can reduce not only the programming effort but also the cost of development for creating geospatial data visualizations (*i.e.*, markedmaps, clusteredmaps and heatmaps)**(Validades H2)**.

Final Remarks - Research Questions

Q2 - *Can the parallel code generated by this DSL speed up the data processing of raw geospatial data?*

- The GMaVis compiler implementation demonstrates that it is possible to generate SPar annotations automatically to speed up the data processing **(Validades H3)**.
- Results demonstrate that both versions (SPar and TBB) increased performance by decreasing the execution time when compared to the sequential version, but SPar required less code generation **(Validades H4)**.

Limitations

- Shared memory architectures **have a bottleneck when reading data from disk**. First version of GMaVis was implemented targeting these types of architectures.

Limitations

- Shared memory architectures **have a bottleneck when reading data from disk**. First version of GMaVis was implemented targeting these types of architectures.
- **Low expressiveness**. This is because to offer a higher level of abstraction, it is required to keep some features hidden from users that allow changing some visualization details, such as colors, icons, and specific sizes.

Limitations

- Shared memory architectures **have a bottleneck when reading data from disk**. First version of GMaVis was implemented targeting these types of architectures.
- **Low expressiveness**. This is because to offer a higher level of abstraction, it is required to keep some features hidden from users that allow changing some visualization details, such as colors, icons, and specific sizes.
- This DSL has **focused in geospatial data visualizations**. However, data analysis may require the use of different visualization types/techniques to achieve its objective.

Limitations

- Shared memory architectures **have a bottleneck when reading data from disk**. First version of GMaVis was implemented targeting these types of architectures.
- **Low expressiveness**. This is because to offer a higher level of abstraction, it is required to keep some features hidden from users that allow changing some visualization details, such as colors, icons, and specific sizes.
- This DSL has **focused in geospatial data visualizations**. However, data analysis may require the use of different visualization types/techniques to achieve its objective.
- The data preprocessor in GMaVis only supports **non-hierarchical** data files.

Future Work

- Improve this DSL to perform in **distributed memory architectures**.

Future Work

- Improve this DSL to perform in **distributed memory architectures**.
- The insertion and support of **other visualization types** is planned for the future, using geospatial data.

Future Work

- Improve this DSL to perform in **distributed memory architectures**.
- The insertion and support of **other visualization types** is planned for the future, using geospatial data.
- The creation of a pre-parser for **hierarchical files** are planned for future work. Also, it is possible to use or convert a hierarchical into a non-hierarchical data file.

Future Work

- Improve this DSL to perform in **distributed memory architectures**.
- The insertion and support of **other visualization types** is planned for the future, using geospatial data.
- The creation of a pre-parser for **hierarchical files** are planned for future work. Also, it is possible to use or convert a hierarchical into a non-hierarchical data file.
- Data declarations can be extended in this interface to offer **smart classifications and data selection** through the use of machine learning and data mining algorithms.

Published Papers

Related to this work

- **Ledur, C.**; Griebler, D.; Fernandes, L. G.; Manssour, I. "Uma Linguagem Específica de Domínio com Geração de Código Paralelo para Visualização de Grandes Volumes de Dados". In: Escola Regional de Alto Desempenho (ERAD), 2015, pp. 2.
- **Ledur, C.**; Griebler, D.; Manssuor, I.; Fernandes, L. G. "Towards a Domain-Specific Language for Geospatial Data Visualization Maps with Big Data Sets". In: ACS/IEEE International Conference on Computer Systems and Applications, 2015, pp. 8.
- (SUBMISSION) **Ledur, C.**; Griebler, D.; Fernandes, L. G.; Manssour, I. "GMaVis: A Domain-Specific Language for Geospatial Data Visualizations". In: IEEE Information Visualization (InfoVis), IEEE VIS, 2016

Published Papers

Contributions

- Adornes, D.; Griebler, D.; **Ledur, C.**; Fernandes, L. G. "A Unified MapReduce Domain-Specific Language for Distributed and Shared Memory Architectures". In: The 27th International Conference on Software Engineering & Knowledge Engineering, 2015, pp. 6.
- Adornes, D.; Griebler, D.; **Ledur, C.**; Fernandes, L. G. "Coding Productivity in MapReduce Applications for Distributed and Shared Memory Architectures", International Journal of Software Engineering and Knowledge Engineering, 2015, pp. 4.

References I

S. K. Card, J. D. Mackinlay, and B. Shneiderman.
Readings in Information Visualization: Using Vision to Think.
Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999.

C. Chiw, G. Kindlmann, J. Reppy, L. Samuels, and N. Seltzer.
Diderot: A Parallel DSL for Image Analysis and Visualization.
In *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation*, volume 47 of *PLDI'12*, pages 111–120, New York, USA, June 2012. ACM.

H. Choi, W. Choi, T. Quan, D. G. Hildebrand, H. Pfister, and W.-K. Jeong.
Vivaldi: A Domain-Specific Language for Volume Processing and Visualization on Distributed Heterogeneous Systems.
IEEE Transactions on Visualization and Computer Graphics, 20(12):2407–2416, 2014 December.

D. Griebler.
A New Compiler-Based Framework Perspective for High-Level Stream Parallelism.
PhD thesis, Faculdade de Informática - PPGCC - PUCRS, Porto Alegre, Brazil, March 2016.

M. Hasan, J. Wolfgang, G. Chen, and H. Pfister.
Shadie: A Domain-Specific Language for Volume Visualization, 2010 (accessed December 15, 2015).

C. Ledur, D. Griebler, I. Manssuor, and L. G. Fernandes.
Towards a Domain-Specific Language for Geospatial Data Visualization Maps with Big Data Sets.
In *ACS/IEEE International Conference on Computer Systems and Applications*, AICCSA'15, page 8, Marrakech, Marrocos, November 2015. IEEE.

References II

L. A. Meyerovich, M. E. Torok, E. Atkinson, and R. Bodík.
Superconductor: A Language for Big Data Visualization, February 2013 (accessed December 17, 2015).

P. Rautek, S. Bruckner, M. Groller, and M. Hadwiger.
ViSlang: A System for Interpreted Domain-Specific Languages for Scientific Visualization.
IEEE Transactions on Visualization and Computer Graphics, 20(12):2388–2396, December 2014.

Voltar para Capa

Questions

Questions & Answers

Thank you

Thank you!

More questions:

cleverson.ledur@acad.pucrs.br
cleversonledur@gmail.com
http://gmap.pucrs.br/cleversonledur/