RawVis: A System for Efficient In-situ Visual Analytics

Stavros Maroulis
Nat. Techn. Univ. of Athens &
ATHENA Research Center, Greece

Nikos Bikakis
ATHENA Research Center, Greece

George Papastefanatos
ATHENA Research Center, Greece

Panos Vassiliadis
University of Ioannina, Greece

Yannis Vassiliou
Nat. Techn. Univ. of Athens, Greece

ABSTRACT

In-situ processing has received a great deal of attention in recent years. In in-situ scenarios, big raw data files which do not fit in main memory, must be efficiently handled on-the-fly using commodity hardware, without the overhead of a preprocessing phase or the loading of data into a database system. This paper presents RawVis, an open source data visualization system for in-situ visual exploration and analytics over big raw data. RawVis implements novel indexing schemes and adaptive processing techniques allowing users to perform efficient visual and analytics operations directly over the data files. RawVis provides real-time interaction, reporting low response time, over large data files, using commodity hardware.

KEYWORDS

Big Raw Data, Visualization Tool, Adaptive and Progressive Indexing, Big Data Visualization, Data Exploration, Visual Analytics, RawVis, Interactive Query Processing

ACM Reference Format:
Stavros Maroulis, Nikos Bikakis, George Papastefanatos, Panos Vassiliadis, and Yannis Vassiliou. 2021. RawVis: A System for Efficient In-situ Visual Analytics. In *Proceedings of the 2021 International Conference on Management of Data (SIGMOD '21), June 20–25, 2021, Virtual Event, China.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3448016.3452764

1 INTRODUCTION

Commonly, in data exploration scenarios, users wish to *visually interact and analyze* large data files that *do not fit in main memory*, e.g., data produced by scientific workflows, IoT devices or crowdsourcing. In several cases, these users usually have limited skills in data management and processing, as well as *limited resources or commodity hardware for use*, e.g., not a distributed environment. Ideally, the tasks in such scenarios, require a very small *raw data-to-analysis time* and *memory resources*, as well as *efficient visual exploration and analytic operations*, which will be performed via interactive visualizations and analytical methods, without being overwhelmed in the tedious processes of data loading, indexing and query optimization [3].

SIGMOD '21, June 20–25, 2021, Virtual Event, China

ACM ISBN 978-1-4503-8343-1/21/06…$15.00
https://doi.org/10.1145/3448016.3452764

In the last few years, the *in-situ paradigm* has been adopted in the context of data exploration scenarios, referring to the collection of data access methods that enable the on-the-fly analysis over large sets of raw data, i.e., data files in raw formats like CSV or JSON [1, 9, 13, 14, 18, 19, 24]. In-situ techniques attempt to avoid the overhead of fully loading and indexing the data in a DBMS and improve performance by progressively building an index as the user explores data.

Contribution. To address the aforementioned challenges, we propose the RawVis system[1] that enables in-situ visual exploration and analytics over large raw data files, for users who wish to make use of commodity hardware [4, 5, 15]. RawVis uses innovative indexing schemes and adaptive techniques, which allow users to perform visual and analytic operations directly over the raw data, without the need of an underlying DBMS or a query engine. As mentioned later, RawVis exhibited low response time over large datasets (e.g., 50G, 100M objects) using commodity hardware, which in turn allows its adoption by interactive applications. Note that, in most cases, RawVis is noticeably faster and performs significantly fewer I/O operations compared to various competitive systems [4, 5, 15].

In our working scenario, the user selects a raw file to visualize and analyze, the file is parsed and indexed on-the-fly, generating a "crude" initial version of our index. The user, then, performs visual operations, which are translated to queries evaluated over the index. Based on the user interaction, the index is adapted incrementally, adjusting its structure and updating statistics.

Demonstration. The graphical user interface of RawVis allows the audience to perform on-the-fly visual exploration and analytical operations over large raw data. The interface offers numerous features, such as: map-based exploration, statistics computation, data analysis via several visual methods. Datasets from two different domains (telecommunication and transport) will be available as the use cases through which users can explore, analyze and test the capabilities of the tool.

Related Work. Relevant works in this area have proposed techniques for progressive loading and indexing of the data, for generic in-situ querying [1, 9, 13, 14, 18, 19, 24], without focusing on the specific needs of *visualization and low-latency interaction* with raw data. Instead, our work considers the in-situ processing as a sequence of query operations performed by the user in the context of spatial exploration and visual analysis of data residing in raw files.

Moreover, in the context of in-situ processing, in contrast to our work, there is no work studying *exploratory queries combined with aggregate operations over categorical attributes* i.e., group-by queries

[1] The source code is available at: https://github.com/VisualFacts/RawVis

that filter and aggregate results based on categorical values. The Group-by operation *is essential in order to generate the most-known visualization types*, in which categorical-based aggregated results are visualized, such as: bar charts, heatmaps, parallel coordinates, (binned) scatter plots, radar chart, pies. Beyond the Group-by operation, the Filter operation over categorical attributes, enables the support of effective exploration mechanisms, e.g., faceted search.

In our experiments [4, 5, 15], RawVis is in most cases about 5-10× *faster* than the competitors, and requires significantly less *memory resources*. Particularly, in the case of categorical-based aggregate operations, RawVis is about 40× *faster* and performs up to 3 *orders of magnitude fewer* I/O *operations*, compared to existing solutions. It is worth noting that RawVis exhibits in most cases *response time less than* 1*sec*, over large raw file (e.g., 50G, 100M objects) using commodity hardware.

Moreover, several works study the problem of incremental and adaptive indexes [2, 6–8, 10–12, 16, 17, 20–23]. However, in these works the data has to be previously loaded in the system. Hence, they are not applicable on in-situ query scenarios.

2 SYSTEM OVERVIEW

Figure 1 presents the architecture of the RawVis system; the frontend (web-based) is presented on the left side of the figure, and the backend on the right.

Exploration Scenario. In our working scenario, we consider that a *user visually explores* the data stored in a single *csv data file* Ⓐ in disk using a *2D visualization technique* (e.g., map, scatter plot) Ⓑ, and *analyzes it using visual* (e.g., bar and line charts, heatmaps, parallel coordinates), and *statistical* methods Ⓒ (e.g., Pearson correlation, covariance). Data attributes may be numeric, spatiotemporal, categorical, or textual; at least two of them must be numeric (e.g., longitude, latitude) and can be mapped to the X and Y axis of the 2D visualization.

Architecture Overview. In the backend, ❶ the *first time* the user requests to visualize or analyze a new dataset, the *file is parsed and indexed on-the-fly*, generating a "*crude*" initial version of the index (*Index Initialization* component). In parallel with the index construction, the results corresponding to the first user request are evaluated. ❷ Based on the *exploration model* the user's visual and analytic operations (i.e., interactions) are *translated to data-access operations* (*Operation Translation* component), which are then ❸ evaluated over the *Tile-Tree index* structure (*Query Evaluation* component) to compute and fetch the results. ❹ Based on the last user request, the index is *adapted progressively*, reorganizing its contents, and updating computed statistics (*Index Adaptation* component). ❺ The query results are further processed and reduced (e.g., via clustering, sampling, aggregation) before they are rendered in the visualization component, such that over-plotting issues are properly addressed (*Data Reduction* component). ❻ Finally, the results are returned and visualized to the user. Note that, during the index construction or the query evaluation, the index structure may not fit in main memory; in such cases, the *Eviction Handler* component stores parts of the index structure in the disk.

In the frontend, a 2D visual data representation is presented, as well as exploration operations, visual analytics and statistics (Sect. 3).

Exploration Model. RawVis is based on a formal exploration model that defines a set of *exploratory and analytic operations* which *formulate the user interactions*. RawVis offers functionality for the user to explore and analyze the underlying data objects over a 2D plane as well as through various ad-hoc charts and statistics. It implements basic exploration operations over a 2D visualization plane, such as pan, zoom, filter, view object details, and selection of objects by defining arbitrary areas over the visualized objects. Beyond exploratory operations, the model defines analytic operations which allow the user to visually analyze the objects by generating several visualization types that can aggregate, compare or provide statistics on one or more data properties. For example, group by and aggregate visualized objects in bar charts, generate a heatmap that depicts the correlation between attributes, or compute statistics over object attributes. These operations are translated to data-access operators, forming a query applied to the index, denoted as *exploratory query* (or *query*).

Indexing Scheme. The indexing scheme is designed in the context of visual exploration based on the following basic principles: (1) fast on-the-fly construction (see next paragraph); (2) memory and I/O-efficiency; and (3) efficient statistics computations.

RawVis is built on top of a *lightweight main memory* indexing scheme which combines: a *tile-based multilevel structure* organizing data into tiles for efficient exploration in the 2D plane; with a *tree-based structure* which organizes a tile's objects based on its categorical values, offering efficient categorical-based group-by and filter operations. Particularly, the tile structure organizes the data objects into hierarchies of non-overlapping rectangle tiles, which are defined over the axis attributes of the objects. Moreover, each leaf tile is associated with a tree that organizes the objects enclosed in its associated tile based on categorical attribute values.

Statistics computations are performed using *metadata* stored in the tree nodes. Metadata are basically numeric values calculated by algebraic aggregate functions over one or more attributes of the objects. The structure of the tree, allows efficient statistics computations over different attributes, by performing in-memory aggregate operations.[2]

Regarding the I/O operations, the proposed indexing scheme enables the access of the raw file in a sequential manner, which noticeably reduces the cost of I/Os. The statistics computations and the cost of I/O operations, are further improved by the index adaptation (more details in the next paragraphs).

Index Initialization. During the initialization phase, the characteristics of the index are determined (e.g., tile indexing precision, number and size of trees, tree's nodes ordering); then, during the parsing of the file, an initial version of the index is constructed and the first query is evaluated.

[2]Note that, the support of algebraic aggregate functions in our scheme, enables the computation of a very large number of statistics. Particularly, more than 90% and 75% of the statistics supported by SciPy and Wolfram, respectively, are defined as algebraic aggregate functions [25].

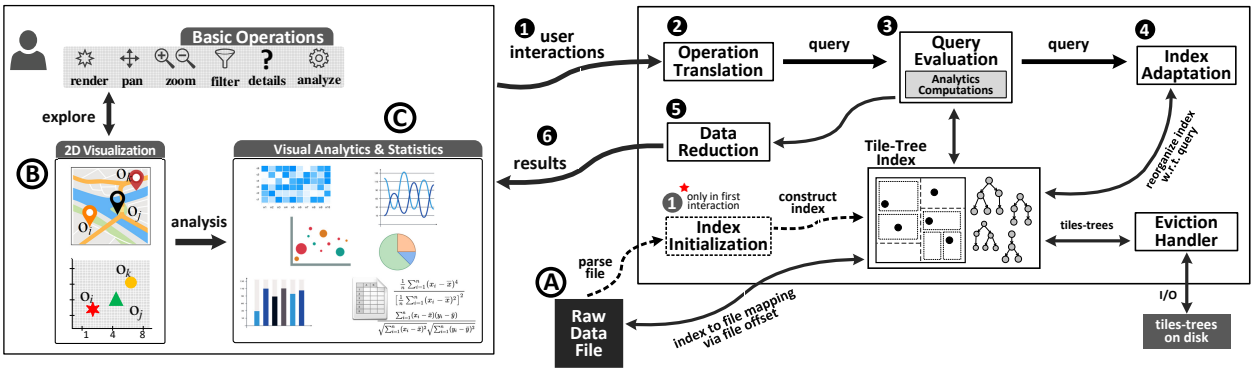

Figure 1: RawVis Architecture

The initialization phase has to handle three major challenges. First, due to our interactive exploration scenario, the *index construction should be performed as fast as possible* in order to entail a small overhead in the evaluation of the first query. Second, the *memory requirements should be relatively small*, since commodity hardware may be used (e.g., laptop). Third, the index initialization, should *improve query evaluation during the initial stages of the exploration*, when the index is not fully adapted yet. Thus, initially, a "crude", lightweight initial version of the index is built, where its characteristics are determined by the examined factors, such as the aforementioned challenges to be handled.

Progressive Index Adaptation. RawVis employs a progressive index adaptation technique that attempts to reduce the I/O operations by adjusting the index based on the user interactions, e.g., exploration areas and categorical attributes under analysis. Index adaptation is performed by modifying the structure of the index (e.g., tile size, tree node ordering); and by enriching and updating its "information", i.e., indexing new attributes, computing node statistics.

The adaptation method incrementally splits the tiles that overlap with a query into smaller subtiles. The splitting process considers factors related to I/O cost in order to decide whether to perform a split or not. Considering the locality-based characteristics of the exploration scenarios, tile splitting increases the likelihood that a future query will fully overlap a tile in the area that the user exploration focuses. The case of fully overlapped tiles allows the index to use the existing metadata, improving the query performance by reducing I/O operations on the file. In conjunction with the tile splitting, the index may be enriched by including new categorical values in the tree structures or by computing different metadata.

During the adaptation of the index, the objects are reorganized in the new tiles and tree structures, and their metadata is updated. The new tree structures (resp. metadata) may be constructed (resp. computed) from scratch, or extended by exploiting the existing ones.

Implementation Details. RawVis is implemented on top of several open-source tools and libraries and is available under GNU/GPL.[1] The frontend was developed in TypeScript as a single-page application using the React library, while the backend was developed in Java 1.8. The frontend client app interacts with the backend with a REST API. For the visualization of the results the Leaflet and Highcharts libraries were used.

3 RawVis USER INTERFACE

This section briefly introduces the RawVis visual interface (Fig. 2). The tool is available at: *http://rawviz.imsi.athenarc.gr*. Also a video presenting the basic functionality of our prototype is available at: *https://vimeo.com/500596816*.

Figure 2 depicts pick-up points from the NYC Yellow Taxi Trip dataset[3], which is CSV files, containing information regarding yellow taxi rides in NYC. Each object refers to a specific taxi ride described by several attributes, such as pick-up location, trip distance, payment type, passenger count, tip amount.

UI Panels. The UI consists of the following panels. The *Dataset Infomation panel* Ⓐ allows the user to select the CSV file to explore, and presents its details, such as the axis attributes (latitude and longitude) that will be used for the map-based exploration. Selecting a different CSV file loads details about its specific data attributes and updates the rest of the UI components. The *Map Visualization panel* Ⓑ offers map-based exploration, allowing the user to interact with the map (e.g., zoom, pan). The *Area Selection panel* Ⓒ allows the user to select an area over the map by drawing a rectangle. The *Filtering panel* Ⓓ provides filtering operations over categorical attributes, allowing the user to perform faceted exploration. The *Statistics panel* Ⓔ presents statistics regarding the selected objects, e.g., univariate statistics such as correlation, standard deviation, and bivariate statistics such as Pearson correlation. The *Analysis panel* Ⓕ allows the user to perform data analysis tasks by selecting visualization type (e.g., bar chart, heatmap, pie), data attributes and aggregate function.
Next, we outline the basic features provided by the RawVis interface.

Map-based Visual Exploration. The user is able to explore, analyze and compare data over different geographical areas using operations like panning and zooming. For example, they may navigate and compute statistics over two different cities. Also, the user can focus on a specific area (e.g., neighborhood) by drawing a rectangle over the map Ⓒ.

[3]https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page

Figure 2: RawVis User Interface

Faceted Exploration. Faceted exploration and analysis is supported by allowing the user to define multiple filters over the categorical attributes via the Filtering panel Ⓓ. For example, in Figure 2, the user has selected to analyze the taxi trips which have been paid using a credit card (Payment type is CRD).

Statistics Computations. The interface offers the user the ability to analyze the data, through the statistics panel Ⓔ. Particularly, the user can examine univariate (e.g., mean, variance, standard deviation) or bivariate statistics (e.g., the Pearson correlation, covariance) over the data attributes. In Figure 2, univariate statistics have been computed for the fare amount, and bivariate for the fare and tip amount.

Visual Analysis. The user is able to visually analyze the data by selecting the most suitable visualization type and metrics to accomplish their analysis Ⓕ. Particularly, the user can select one or more variables to analyze, as well as the visualization type and metric. For example, in Figure 2, the user has selected a bar char to visualize the average tip value w.r.t. the payment method (e.g., cash, credit card). In the second case, the user has selected a heatmap, to visualize the average taxi fare per passenger count and payment type.

4 DEMONSTRATION OUTLINE

In this section, we outline our demonstration scenario. The attendees will be able to interact with RawVis and analyze real-world datasets from several domains e.g., telecommunication and transport. Some of the demonstrated datasets that will be used are: (1) the NYC Yellow Taxi Trip Records dataset; (2) an anonymized telecommunication dataset containing signal strength and latency measurements, where each measurement contains data regarding the geographic location, signal strength, latency, network bandwidth, device brand, network provider, and network technology.

Initially, users will be presented with the various components and operations supported in RawVis. Then, they will be able to interact with the prototype by performing operations such as:

- Select a dataset to explore.
- Interact with the map to pan, zoom in/out to find areas of interest.
- Filter the objects using the facet-based browsing panel.
- Focus on and analyze specific areas by using the rectangle selection functionality.
- Select the statistics to examine during the exploration.
- Select the visualization type, the attributes and the metrics that will be generated in order to support their analysis tasks.

Users will also be presented with specific use case scenarios and exploration tasks that will provide better insight into RawVis visual analytic capabilities.

In the case of the NYC Taxi dataset, users will explore different areas in New York and get insights such as: the most common payment method, the average tip amount, the correlation between the tip and the fare values, how the number of passengers and the payment method affects the amount of the tip, or if a taxi ride starting from the airport results in higher tip amount.

In the use case of the telecommunication dataset, some of the user tasks may be: to identify areas where a large number 5G measurements have been reported; examine which telecommunication providers have better signal in some areas; try to identify if there is correlation between the signal strength and bandwidth; or examine the average signal strength per provider and network technology.

Acknowledgment. This work is funded by the project VisualFacts (#1614 - 1st Call of the Hellenic Foundation for Research and Innovation Research Projects for the support of post-doctoral researchers).

REFERENCES

[1] I. Alagiannis, R. Borovica, M. Branco, S. Idreos, and A. Ailamaki. Nodb: Efficient Query Execution on Raw Data Files. In *ACM International Conference on Management of Data (SIGMOD)*, 2012.

[2] K. Alexiou, D. Kossmann, and P. Larson. Adaptive Range Filters for Cold Data: Avoiding Trips to Siberia. *VLDB Endowment*, 6(14), 2013.

[3] N. Bikakis. Big Data Visualization Tools. In *Encyclopedia of Big Data Technologies*. Springer, 2019.

[4] N. Bikakis, S. Maroulis, G. Papastefanatos, and P. Vassiliadis. RawVis: Visual Exploration over Raw Data. In *Conference on Advances in Databases and Information Systems (ADBIS)*, 2018.

[5] N. Bikakis, S. Maroulis, G. Papastefanatos, and P. Vassiliadis. In-situ visual exploration over big raw data. *Information Systems, Elsevier*, 95, 2021.

[6] G. Graefe and H. A. Kuno. Self-selecting, self-tuning, incrementally optimized indexes. In *International Conference on Extending Database Technology (EDBT)*, 2010.

[7] F. Halim, S. Idreos, P. Karras, and R. H. C. Yap. Stochastic Database Cracking: Towards Robust Adaptive Indexing in Main-Memory Column-Stores. *VLDB Endowment*, 5(6), 2012.

[8] P. Holanda, S. Manegold, H. Mühleisen, and M. Raasveldt. Progressive Indexes: Indexing for Interactive Data Analysis. *PVLDB*, 12(13), 2019.

[9] S. Idreos, I. Alagiannis, R. Johnson, and A. Ailamaki. Here Are My Data Files. Here Are My Queries. Where Are My Results? In *Conference on Innovative Data Systems Research (CIDR)*, 2011.

[10] S. Idreos, M. L. Kersten, and S. Manegold. Database Cracking. In *Conference on Innovative Data Systems Research (CIDR)*, 2007.

[11] S. Idreos, M. L. Kersten, and S. Manegold. Self-organizing tuple reconstruction in column-stores. In *ACM International Conference on Management of Data (SIGMOD)*, 2009.

[12] S. Idreos, S. Manegold, H. A. Kuno, and G. Graefe. Merging What's Cracked, Cracking What's Merged: Adaptive Indexing in Main-Memory Column-Stores. *VLDB Endowment*, 4(9), 2011.

[13] M. Karpathiotakis, I. Alagiannis, and A. Ailamaki. Fast Queries Over Heterogeneous Data Through Engine Customization. *VLDB Endowment*, 9(12), 2016.

[14] M. Karpathiotakis, M. Branco, I. Alagiannis, and A. Ailamaki. Adaptive Query Processing on Raw Data. *VLDB Endowment*, 7(12), 2014.

[15] S. Maroulis, N. Bikakis, G. Papastefanatos, P. Vassiliadis, and Y. Vassiliou. Adaptive indexing for in-situ visual exploration and analytics. In *ACM International Workshop on Data Warehousing and OLAP (DOLAP)*, 2021.

[16] V. Nathan, J. Ding, M. Alizadeh, and T. Kraska. Learning multi-dimensional indexes. In *ACM International Conference on Management of Data (SIGMOD)*, 2020.

[17] M. Nerone, P. Holanda, E. C. de Almeida, and S. Manegold. Multidimensional Adaptive and Progressive Indexes. In *IEEE International Conference on Data Engineering (ICDE)*, 2021.

[18] M. Olma, M. Karpathiotakis, I. Alagiannis, M. Athanassoulis, and A. Ailamaki. Slalom: Coasting through Raw Data Via Adaptive Partitioning and Indexing. *VLDB Endowment*, 2017.

[19] M. Olma, M. Karpathiotakis, I. Alagiannis, M. Athanassoulis, and A. Ailamaki. Adaptive partitioning and indexing for in situ query processing. *VLDBJ*, 2019.

[20] M. Pavlovic, D. Sidlauskas, T. Heinis, and A. Ailamaki. QUASII: query-aware spatial incremental index. In *International Conference on Extending Database Technology (EDBT)*, 2018.

[21] M. Pavlovic, E. Tzirita Zacharatou, D. Sidlauskas, T. Heinis, and A. Ailamaki. Space odyssey: efficient exploration of scientific data. In *ExploreDB*, 2016.

[22] E. Petraki, S. Idreos, and S. Manegold. Holistic Indexing in Main-memory Column-stores. In *ACM International Conference on Management of Data (SIGMOD)*, 2015.

[23] S. Richter, J. Quiané-Ruiz, S. Schuh, and J. Dittrich. Towards zero-overhead static and adaptive indexing in Hadoop. *Journal on Very Large Data Bases (VLDBJ)*, 23(3), 2014.

[24] Y. Tian, I. Alagiannis, E. Liarou, A. Ailamaki, P. Michiardi, and M. Vukolic. Dinodb: An Interactive-speed Query Engine for Ad-hoc Queries on Temporary Data. *IEEE Transactions on Big Data*, 2017.

[25] A. Wasay, X. Wei, N. Dayan, and S. Idreos. Data Canopy: Accelerating Exploratory Statistical Analysis. In *ACM International Conference on Management of Data (SIGMOD)*, 2017.