Faster Visual Analytics through Pixel-Perfect Aggregation

Uwe Jugel, Zbigniew Jerzak,
Gregor Hackenbroich
SAP SE
Chemnitzer Str. 48, 01187 Dresden, Germany
{firstname}.{lastname}@sap.com

Volker Markl
Technische Universität Berlin
Straße des 17. Juni 135
10623 Berlin, Germany
volker.markl@tu-berlin.de

ABSTRACT

State-of-the-art visual data analysis tools ignore bandwidth limitations. They fetch millions of records of high-volume time series data from an underlying RDBMS to eventually draw only a few thousand pixels on the screen.

In this work, we demonstrate a pixel-aware big data visualization system that dynamically adapts the number of data points transmitted and thus the data rate, while preserving pixel-perfect visualizations. We show how to carefully select the data points to fetch for each pixel of a visualization, using a visualization-driven data aggregation that models the visualization process. Defining all required data reduction operators at the query level, our system trades off a few milliseconds of query execution time for dozens of seconds of data transfer time. The results are significantly reduced response times and a near real-time visualization of millions of data points.

Using our pixel-aware system, the audience will be able to enjoy the speed and ease of big data visualizations and learn about the scientific background of our system through an interactive evaluation component, allowing the visitor to measure, visualize, and compare competing visualization-related data reduction techniques.

1. INTRODUCTION

High-volume time series data are ubiquitous in many domains, such as finance, discrete manufacturing [6], or sports analytics [8]. It is not uncommon that millions of readings from high-frequency sensors are subsequently stored in relational database management systems (RDBMS), to be later accessed using visual data analysis tools.

Modern data analysis tools must support a "fluent and flexible use of visualizations"[4] and still be able to "squeeze a billion records into a million pixels"[10]. In this regard, one open issue for the database community is the development of "compact data structures that support algorithms for rapid data filtering, aggregation, and display rendering"[10].

Unfortunately, these issues are yet unsolved for existing RDBMS-based visual data analysis tools, such as Tableau Desktop 8.1 (tableausoftware.com), SAP Lumira 1.13 (saplumira.com), QlikView 11.20 (clickview.com), or Datawatch Desktop 12.2 (datawatch.com). While they provide flexible and direct access to relational data sources, they do not consider an automatic, visualization-related data filtering or aggregation and are not able to quickly and easily visualize high-volume time series data, having 1 million records or more. For example, as illustrated in Figure 1a, they redundantly store copies of the raw data as tool-internal objects, requiring significant amounts of system memory per record. This causes long waiting times for the users, leaving them with unresponsive tools or even impairing the user's operating systems, in case the system memory is exhausted.

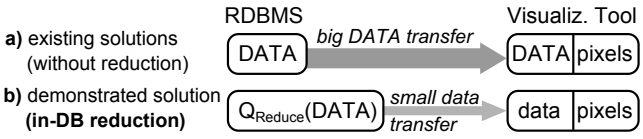

Figure 1: Data transfer in visual analytics tools.

However, the final visualization of any high-volume data set is inherently restricted by a *width* × *height* pixel matrix. Any visualization tool must eventually rescale and rasterize the data to this limited amount of pixels. In previous work [7], focusing on line charts, we showed how to model this implicit data reduction as database queries, driven only by the *width* of the line chart.

In this work, we extend this pixel-aware data aggregation approach to other chart types and demonstrate it in an end-to-end visualization system together with the data analytics tool SAP Lumira. Driven by the *type*, *width*, and *height* of a visualization, we model the data reduction by means of query rewriting, using only the relational algebra and the common aggregation functions: *min*, *max*, and *avg*. Consequently, our system conducts all data reduction directly inside the database, as illustrated in Figure 1b.

Our Lumira-based demonstrator not only provides the audience with an end-to-end visualization system, gracefully handling the interactive visualization of millions of rows of time series data (cf. Figure 2), but also with an interactive monitoring component to facilitate a live evaluation of our

Proceedings of the VLDB Endowment, Vol. 7, No. 13

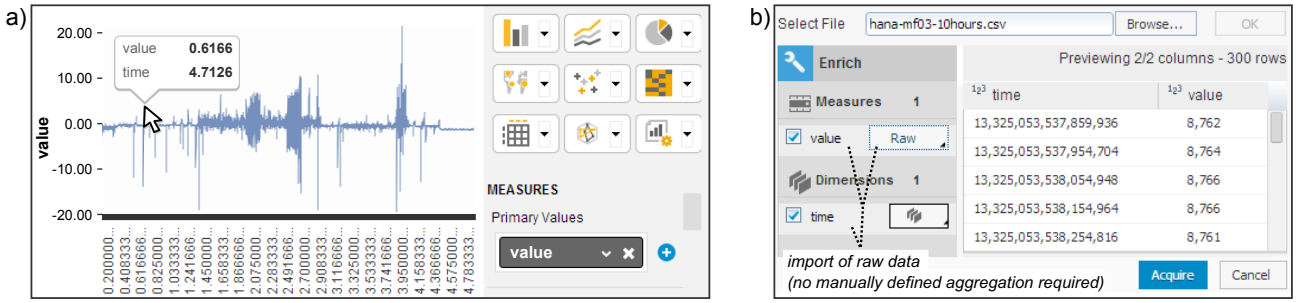

Figure 2: a) Displaying 5 Million rows in SAP Lumira. b) Import of high-volume time series data set.

approach. The monitor supports inspecting and comparing data-reduced and original queries, depicting pixel-level differences of the resulting visualizations, and displaying extensive runtime information like query execution times. Using our interactive demonstrator, the visitors can learn how pixel-aware data aggregation provides pixel-perfect visualizations, as derivable from the original data, while still achieving data reduction rates of up to several orders of magnitude.

2. SOLUTION OVERVIEW

Figure 3 illustrates our system architecture. The *visualization client* (Lumira UI) issues a visualization-related *query* to a server-side query interface (Lumira Server). In our enhanced version of Lumira, these queries are intercepted and augmented by a *query rewriter* before being executed. Depending on the *type*, the width w, and the height h of the visualization, we rewrite the original query with additional aggregation-based data reduction operators (cf. Section 2.2).

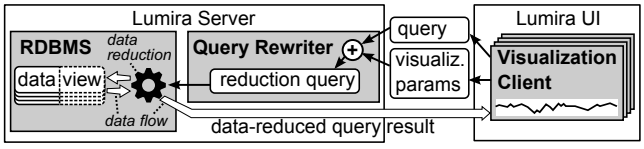

Figure 3: Visualization-driven query rewriting.

2.1 Visualization Model

There exist dozens of ways to visualize numerical data. However, for visualizing high-volume time series data, we only consider common visualizations that consume but a few pixels per data point [2], i.e., scatter plots and line charts, but also bar charts that have a bar width of 1 pixel.

For these types of visualization, we consider a time series as a relation $T(t, v)$ with a timestamp $t \in \mathbb{R}$ and a value $v \in \mathbb{R}$. A time series with multiple attributes can be represented by two separate time series with a shared time axis in the visualization, i.e., two *aligned time series*. A visualization-related query must comply to this time series model, i.e., project two numerical attributes from the underlying data. Therefore, Lumira allows selecting a numerical measure and a dimension, to create a compatible analytical *view* on the relational *data* (cf. Figure 3).

Other common chart types can be derived from the considered base types. For example, a bubble chart is in fact a scatter plot with varying size of the marks (circles), used for each data point, depending on an additional attribute. The underlying data of a bubble chart can be represented by two *aligned* time series relations. Similarly, stacked line or bar charts can be modeled as combination of two aligned line or bar charts. Focusing on time series visualizations, we currently exclude chart types that do not have a distinct time axis, e.g., tree maps, heat maps, pie charts, etc.

2.2 Pixel-Aware Data Aggregation

A time series visualization displays the relation of two numerical attributes time t and value v in 2D space, using $w \times h$ screen pixels. An appropriate data reduction for each type of visualization depends on how each tuple (t, v) is presented on the screen, i.e., which pixels it occupies. The considered chart types have the following underlying rendering properties and resulting data reduction operators.

Scatter Plots use small markers placed in 2D space, occupying one or several neighboring pixels. To reconstruct all pixels from a reduced data set, at least one data point per pixel must be selected from the base data. A scatter plot can display up to an upper bound of $n_{max} = w \cdot h$ distinct data points. We can model a corresponding data reduction by grouping the timestamps into w groups and the values into h groups, resulting in one group per pixel. We then compute the average time and value per group (cf. Figure 4c), yielding a time series relation with up to $w \cdot h$ average tuples. We denote this data reduction as *G2D* – grouping in 2D space.

Bar Charts use rectangles, starting at the bottom of the chart and reaching to the *maximum* value of the corresponding time span. A simple bar chart can display at most $n_{max} = w$ distinct data points in 2D space. Multiple values per pixel column (per time span) are overplotted by the *maximum* bar. The corresponding data reduction requires grouping the time into w groups, i.e., grouping by pixel columns, and selecting the tuple with the maximum value per group.

Line Charts use straight line segments, connecting each two consecutive tuples of the underlying time series relation. As illustrated in Figure 4b, line charts can display up to four *distinctly visible*, *non-ambiguous* data points per pixel column (per time span). As a result, line charts can at most display $n_{max} = 4 \cdot w$ distinct data points. A corresponding data reduction requires selecting the *first*, *last*, *min*, and *max* tuples per pixel column (cf. Figure 4b). Note that – contrary to common practice – only selecting the *min* and *max* tuples per pixel column is not sufficient to produce a correct line visualization, as derivable from the original data.

a) **Aggregation Operators**

Chart	*Operator*	*Group By*	n_{max}
scatter plot	**G2D**	each pixel	w*h
bar chart	**Max**	pixel columns	1*w
line chart (with errors)	**Min+Max**	pixel columns	2*w
line chart (error-free)	**M4** (min, max,first,last)	pixel columns	4*w

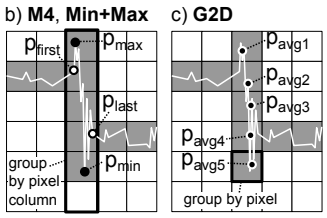

Figure 4: Visualization-driven data aggregation.

We denote this pixel-column-wise selection of the *first*, *last*, *min*, and *max* tuples as *M4 aggregation* [7].

Figure 4a lists the properties of the described visualization-driven aggregation operators, including their maximum number of records n_{max} that needs to be transferred from the database to the visualization client to ensure a correct visualization of the original data. Being able to determine an n_{max} and an appropriate data reduction operator for any considered type of visualization, our system can *automatically* define data reduction queries, without additional user interaction. In contemporary data analytics tools, the user has to *manually* configure presentation-related, time-based aggregations to come up with reasonably small number of records to be displayed.

2.3 Query Rewriting

In many data analytics tools, the user can hover over the geometric primitives to examine detailed information of the underlying data sets. Therefore, line charts are often overlaid with the corresponding scatter plots, so that all data points underlying a line can be inspected. Consequently, our query rewriter composes a query that jointly performs the *G2D* and the *M4* aggregation. This allows drawing pixel-perfect line charts and the inspection of the underlying data, while at most fetching $n_{max} = w \cdot (h + 4)$ tuples from the RDBMS. Figure 5 shows the corresponding SQL query, as a UNION of a *G2D* query and an *M4* query. To define a grouping in 2D space, the *G2D* query scales time t and value v from the original time series relation Q to the visualization's coordinate system and then rounds the rescaled real-valued data to discrete pixels. The *M4* query computes the four extrema and a horizontal group key k per group. The result QA is joined with the original time series Q on the group key and the corresponding aggregated timestamps and values. From the result of this equi-join, the *M4* query then projects the matched tuples (t, v). The final result will then contain up to $w \cdot h$ average tuples from *G2D* and up to $4 \cdot w$ extremum tuples from *M4*.

Our query rewriter intercepts all queries that are used for scatter plots, line charts, bar charts, and other chart types, derivable from these three base types. The considered data reduction is applied to any query that would yield a high-cardinality result. The system behavior of our prototype is as follows. First, we analyze an original query using the EXPLAIN functionality, provided by most RDBMS. If the query would produce a too large result set, i.e., $|Q| > n_{max}$, we wrap it with a data reduction operator. Internally, the actual user query is not changed. We then either execute the original user query or the new data reduction query. The result will always have a predictable cardinality and thus predictable bandwidth requirements. In case we execute the data reduction query, we also indicate that to the user in the Lumira user interface.

```
WITH Q as <time series T(t,v) based on arbitrary query>
SELECT avg(t) as t, avg(v) as v FROM Q --avg. tuple
GROUP BY round($w*(t-$t1)/($t2-$t1)),  --horiz. grouping
         round($h*(v-$v1)/($v2-$v1))   --vert.  grouping
UNION SELECT t,v FROM Q JOIN           --union G2D + M4
(SELECT round($w*(t-$t1)/($t2-$t1)) as k, --horiz. key
        min(t) as t_min, max(t) as t_max  --get 1st,last
        min(v) as v_min, max(v) as v_max, --get min,max
        FROM Q GROUP BY k) as QA          --group by k
ON k = round($w*(t-$t1)/($t2-$t1))        --join on k
   AND (t = t_min OR t = t_max OR         --&(1st|last
        v = v_min OR v = v_max)           -- |min|max)
```

Figure 5: Aggregation-based data reduction query to obtain combined scatter plot and line chart data.

3. RELATED WORK

There are many different approaches for time series dimensionality reduction [3], ranging from simple measures such as piece-wise aggregate approximation (averaging), over random and systematic sampling, to more complex measures, based on line simplification [9, 5, 11]. However, existing techniques drive the data reduction process using geometric measures defined in $\mathbb{R} \times \mathbb{R}$, e.g., using the Euclidean distance of a removed point from an approximating line segment. These generic measures in $\mathbb{R} \times \mathbb{R}$, are not able to catch the rasterization effects caused by the discontinuities at the intersections of pixel rows and pixel columns. What matters for the user is the final visualization having set the correct discrete pixels on the screen. Therefore, our approach uses an aggregation-based data reduction, parametrized by the *width* and *height* of the visualization to emulate the process of rendering real-valued data to discrete screen pixels.

Visualization-Driven Data Reduction has previously been proposed by Burtini et al. [1]. However, in our solution, we further extend the idea towards a really data-centric approach, conducting all data reduction inside the database. We also more precisely incorporate the semantics of line rendering [7], since the common practice of selecting only $1 \cdot width$ tuples is not sufficient to draw error-free line charts (cf. Section 2.2).

4. DEMONSTRATION

In our demonstration, we show how our system automatically and quickly renders visualizations of millions of records. For example, we allow the audience to import millions of sensor data records and visualize them as a line chart, as illustrated by Figure 2. Several such high-volume data sets [6, 8] are preconfigured and available to the audience for exploration and visualization. We also allow the import of custom data sets, via Lumira Cloud (cloud.saplumira.com) or locally via CSV files.

Our demonstrator provides an insight into the technical aspects of our solution. Therefore, we include an interactive query monitor (cf. Figure 6) that allows the audience to

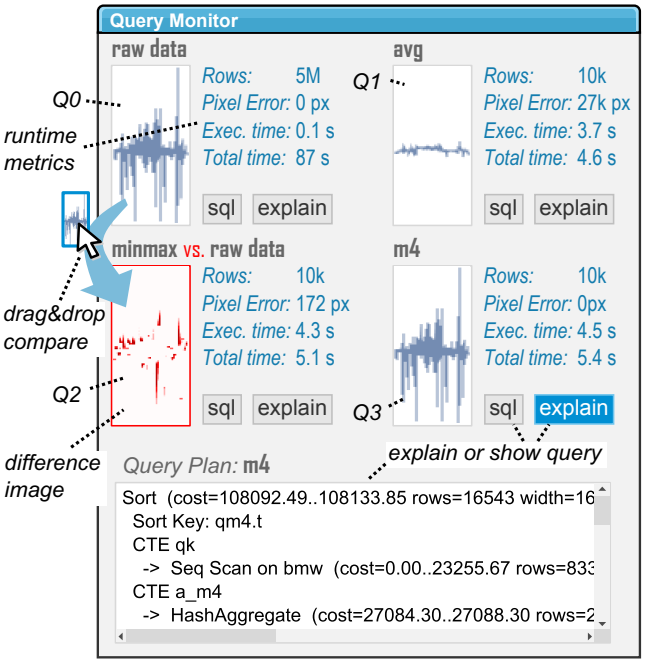

Figure 6: Interactive query monitor.

quickly inspect visualization-related queries and easily compare the visualizations of different query results, based on different data reduction techniques. The monitoring component also shows the corresponding query execution and total response times, query execution plans, measured pixel errors, and the measured structural similarity of a resulting image to the corresponding original image [12]. All derived and original visualizations can be inspected pixel by pixel and compared (via drag & drop) using difference images to let the audience pass their own verdict on the visualization quality.

Using our demonstrator, the audience can observe how visualization-driven data aggregation can provide high quality visualizations at high data reduction rates. For example, the user may select three different pixel-column-wise data aggregations techniques Q1 to Q3 (cf. Figure 6). The user can then compare the performance and visual results of each data reduction technique with the unreduced baseline case (Q0). In this example, all three techniques provide a high data reduction from 5 million to 10k rows, but may produce notable pixel errors. For example, the pixel-column-wise averaging (Q1) flattens the important (vertical) extrema of a time series, resulting in a significantly distorted visualization. The selection of the *min* and *max* tuples per pixel-column (Q2) provides better visual results, but still suffers from pixel errors (depicted by the difference image for Q2 in Figure 6). The viewers can observe that the visual result of our described *M4* aggregation (Q3) always matches perfectly with the baseline visualization (Q0), while requiring only a fraction of the original data to be fetched from the database. Also note that the *total time*, for the user to wait for a query result, is decreased by one order of magnitude, from 87 seconds to only 5.4 seconds.

In summary, our Lumira-based demonstrator allows the audience to analyze preconfigured or custom high-volume time series data sets and evaluate different data reduction approaches, based on the following techniques: averaging, random and systematic sampling, single aggregation (*min*, *max*, *avg*), composite aggregations (combinations of *min*, *max*, *first*, *last*, *avg*), and finally line simplification algorithms [9, 5, 11]. The latter are the most competitive time series dimensionality reduction approaches, though they are still subject to a measurable error, while our aggregation-based technique provides error-free visualizations.

5. CONCLUSION

This paper presented a proof-of-concept version of SAP Lumira that leverages a data-centric and visualization-driven time series dimensionality reduction to facilitate the visualizations of millions of records of time series data. Our demonstrated approach relies only on the relational algebra and conducts any data reduction directly inside the database. Respecting the rendering semantics of the visualizations, specifically of line charts, we demonstrated how our pixel-aware, aggregation-based technique surpasses existing sampling, averaging and line simplification approaches, providing higher visualization quality at competitive data reduction rates.

6. REFERENCES

[1] G. Burtini, S. Fazackerley, and R. Lawrence. Time series compression for adaptive chart generation. In *CCECE*, pages 1–6. IEEE, 2013.

[2] S. G. Eick and A. F. Karr. Visual scalability. *Journal of Computational and Graphical Statistics*, 11(1):22–43, 2002.

[3] T. Fu. A review on time series data mining. *EAAI Journal*, 24(1):164–181, 2011.

[4] J. Heer and B. Shneiderman. Interactive dynamics for visual analysis. *ACM Queue*, 10(2):30, 2012.

[5] J. Hershberger and J. Snoeyink. *Speeding up the Douglas-Peucker line-simplification algorithm.* University of British Columbia, Department of Computer Science, 1992.

[6] Z. Jerzak, T. Heinze, M. Fehr, D. Gröber, R. Hartung, and N. Stojanovic. The DEBS 2012 Grand Challenge. In *DEBS*, pages 393–398. ACM, 2012.

[7] U. Jugel, Z. Jerzak, G. Hackenbroich, and V. Markl. M4: A visualization-oriented time series data aggregation. In *VLDB*. VLDB Endowment, 2014. (submitted, review pending).

[8] C. Mutschler, H. Ziekow, and Z. Jerzak. The DEBS 2013 Grand Challenge. In *DEBS*, pages 289–294. ACM, 2013.

[9] W. Shi and C. Cheung. Performance evaluation of line simplification algorithms for vector generalization. *The Cartographic Journal*, 43(1):27–44, 2006.

[10] B. Shneiderman. Extreme visualization: squeezing a billion records into a million pixels. In *SIGMOD*, pages 3–12. ACM, 2008.

[11] M. Visvalingam and J. Whyatt. Line generalisation by repeated elimination of points. *The Cartographic Journal*, 30(1):46–51, 1993.

[12] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.