Klareco: An Indexing-based Architecture for Interactive Visualization of Heterogeneous Data Sources

Paul Rosen, Alan Morris, Gene Payne, Bill Keach, Ian Walton, Bryony Richards-McClung, John McLennan, Randy Polson, Raymond Levey, Terry Ring, Elizabeth Jurrus, and Greg M. Jones

Abstract— The ETL process (Extract, Transform, and Load) is critical to denormalize data for easy input in visual analysis tools. Unfortunately, this ETL process requires extensive human effort and computation to complete, often spanning months or years before in-depth analysis can be performed. In this paper, we introduce *Klareco*, a visualization architecture that foregoes the ETL process allowing quick access to multiple data sources. The architecture uses an indexing engine for accessing data with multiple schema. A series of small data analysis microservices add intelligence to the architecture. Finally, visualizations are designed to display and explore the data itself, as well as the structure of the data, facilitating discovery. This combination of features enables rapid prototyping of visualizations for a variety of data types, formats, and schema. We demonstrate an early version of the architecture using a case study in the domain of oil and gas exploration and to optimize production.

Index Terms—ETL, data indexing, interactive visualization

1 Introduction

When analyzing data from multiple sources, it is standard practice to engage in the ETL process (Extract, Transform, and Load) to denormalize the data. This process manipulates the data to a consistent structure, making it easy to query for visualization and analysis. For anything beyond the simplest data, this ETL process requires extensive human effort and computation to identify equivalences and relevancies. In large-scale scenarios, the process may span months to years, requiring hundreds of man-hours before the desired information is available for use by the data analyst. It is only after the data are in a common database or data warehouse that analysis can be performed. This means that even the most simple explorations of the data may take days, weeks, months, or even years to complete.

One of the slowest parts of the ETL process is the iterative discovery of important attributes in the data. Analysts will start with a basic denormalization and discover that various pieces of data are missing or unnecessary, requiring reengaging in the ETL process. This comes from two causes. First, before ETL begins, *data analysts don't really understand the structure (schema) of their data*. Second, as new insights are made, *the questions asked by data analysts evolve*. Unfortunately, the time required for each ETL iteration assumes the contrary, hindering the analysts' ability to effectively evolve questions.

We propose a different approach to accessing data for visualization. In this approach, we look for a few specific qualities. First, the solution needs to enable "rapid-vis"—the visualization equivalent of rapid prototyping, where data can be quickly triaged. Second, it must be flexible to a variety of data types, formats, and schema. Finally, the system needs to support some kind of intelligence that replicates functionalities of the ETL process.

To accomplish this goal, we have developed *Klareco*, a three-tiered loosely-couple architecture for accessing multiple data sources. The foundation of the system relies upon an indexing engine for querying the data. While a traditional database query relies strictly upon a well understood database structure (i.e. you query specific tables and fields), indexing is able to relax this requirement. Instead, the indexing engine simply returns a set of records, with a variety of schema, that best match the query terms. The second tier of the system is a set of small data analysis services (microservices) that act as the intelligence of the system, replicating many of the functionalities of the ETL process. These services are used on demand and new services can be easily added. The final tier is the visualization, which must be capable of dealing with the complexity of records with a variety of schema, data types, and missing data. These visualizations must present the data in a way that allows the user to not only see the data but also explore the structure and manipulate the relationships to facilitate further discovery. We believe the combination of these features will enable data analysts to quickly understand their data and adapt to new needs as their analysis questions evolve.

2 Understanding the ETL Process

When new data arrives for analysis, it is almost inevitable that the data is put through the ETL process. The ETL process "massages" data into a form that can be easily loaded into a visualization or analysis tool for investigation. The ETL acronym stands for the three major steps of the process: Extract, Transform, and Load.

- **Extract**: Extract data from sources. These can be homogeneous or heterogeneous sources that are most often, but not always, structured data.
- **Transform**: Modifies the data or the schema in such a way that it can be easily queried by the visualization or analysis tool.
- **Load**: Places the data into a target database for future analysis.

This process produces a stable data structure that can be easily imported into an analysis or visualization tool. This process works particularly well if the data sources to be integrated are small in number and their existing schema are stable.

- *Paul Rosen is with the University of South Florida. E-mail: prosen@usf.edu.*
- *Alan Morris is with the Comprehensive Arrhythmia Research & Management Center at the University of Utah. E-mail: alan.morris@carma.utah.edu.*
- *Gene Payne and Greg Jones are with the Scientific Computing and Imaging Institute at the University of Utah. E-mail: payne.gene@gmail.com and greg@sci.utah.edu.*
- *Bill Keach, Bryony Richards-McClung, John McLennan, Ian Walton, and Raymond Levey are with the Energy and Geosciences Institute at the University of Utah. E-mail: bkeach, brmcclung, jmclennan, iwalton, rlevey@egi.utah.edu.*
- *Randy Polson is with the NanoFab at the University of Utah. E-mail: rcp7@utah.edu.*
- *Terry Ring is with the Department of Chemical Engineering at the University of Utah. E-mail: ring@chemeng.utah.edu.*
- *Elizabeth Jurrus is with the Pacific Northwest National Laboratory. E-mail: liz@sci.utah.edu.*

Manuscript received 31 Mar. 2014; accepted 1 Aug. 2014; date of publication xx xxx 2014; date of current version xx xxx 2014. For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

However, a few major challenges persist with this process. First, it is quite time consuming to design the final database. Not only does one have to wade through a potentially large number of data schema, but the transform stage requires the designer select output fields without necessarily knowing what fields are needed for analysis. This is complicated by the fact that the ETL Engineer and Data Analyst are often different individuals. If chosen incorrectly, the target database needs to be rebuilt. Furthermore, the computation required, mostly in the form of data copying during the load stage, can be time consuming. The final problem is fragility. While issues of data updates can be generally handled gracefully, if the input data schema changes or a dataset is added or removed, the ETL process may need to begin anew.

3 Klareco: An Index-based Visualization Architecture

To address some of these shortcomings, we are developing an indexing-based architecture to quickly deliver data from heterogeneous, structured and unstructured, data sources, regardless of format or storage location, to the visualization system. The architecture, as seen in Figure 1, can be broken down into three main sections.

The first component of the architecture is the indexing engine. The indexing engine acts as a database but removes many of the restrictions of structured query systems (SQL or NoSQL systems)[1]. The indexing engine uses keyword search to access relevant records, with a variety of possible schema, from multiple data sources. Once the data sources are loaded into the index, the system can then be queried, via a Hypertext Transfer Protocol, which returns relevant records in JSON format.

The second component of the architecture is the microservice layer. This layer serves as the intelligence of the system. It's job is to take the mixed-schema data from the indexing engine and output derivatives of that data. These services are intended to be small in size, each producing a very limited and specific type of output. As you will see in our case study (based on oil and gas production data), these functionalities could include date/time, geolocation, oil/gas production, etc. These components communicate in Hypertext Transfer Protocol and JSON.

The final layer is the visualization system. This system is panel-based, where one of many existing visualizations can be selected for viewing the data, or new visualizations can be quickly prototyped and inserted into the software. The challenges to these visualizations, discussed in detail in Section 6, include issues of mixed schema and missing data.

4 Implementation

Our entire system is built using Java allowing it to execute on any platform.

The indexer is based upon Apache Lucene [1], which uses Apache Tika [3] to index a wide variety of data types. The data records are served using the Apache Solr [2] web server. The microservices use Apache Tomcat [4] as a web server. Finally, the visualizations use Java-based Processing [6] system for drawing the visualizations. All components communicate via Hypertext Transfer Protocol, moving data in JSON format.

This architecture represents a very loose coupling between components, meaning that components can be added or substituted quite easily. For example, if a different indexer is desired, it can be easily swapped, so long as the new one understands the same query syntax and outputs JSON data. Another advantage is that components can be written in any programming language. Although we have used Java throughout, there is no reason that a new microservice or visualization application could not be written in C++, Python, etc., so long as it adheres to the correct input/output standards. Finally, the loose coupling of components means that there is no requirement that the indexer, microservice, or visualization components execute on the same machine. Since individual components do not need to be on the same machine, if desired, the local load can be reduced to the visualization only with other components dynamically executed and scaled in a cloud, based upon service need and compute capacity.

[1]While enabling unstructured queries, indexing compared to SQL systems comes with the downside of less precise queries of the data.

5 Case Study: Oil and Gas Exploration

Much of this development work has been done in collaboration with scientists from the Energy and Geoscience Institute (EGI) at the University of Utah. At EGI, scientists are concerned with how a variety of geologic features impact the production of oil and gas through drilling and hydraulic fracturing. Scientists at EGI are actively engaged in interpreting well production data, such as that shown in Figure 3, in terms of reservoir characteristics and drilling, completion and fracturing parameters. The data analysis techniques outlined in this paper will play a critical role in this research. The ultimate goal is to determine if a well should be drilled and what drilling technologies to use

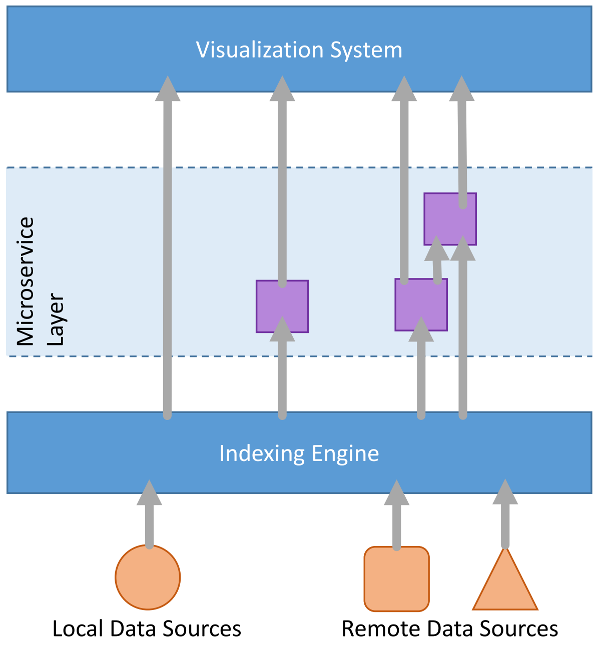

Fig. 1. Illustration of the Klareco Indexing-based Visualization Architecture.

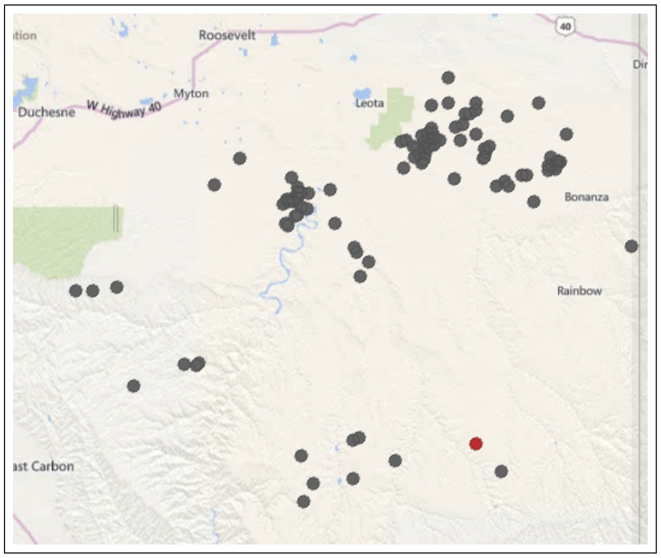

Fig. 2. The first visualization used presents a simple geolocation view of the data. Data is gathered from a special *geolocation* microservice. Keyword-based filtering is easily accomplished by leveraging the index.

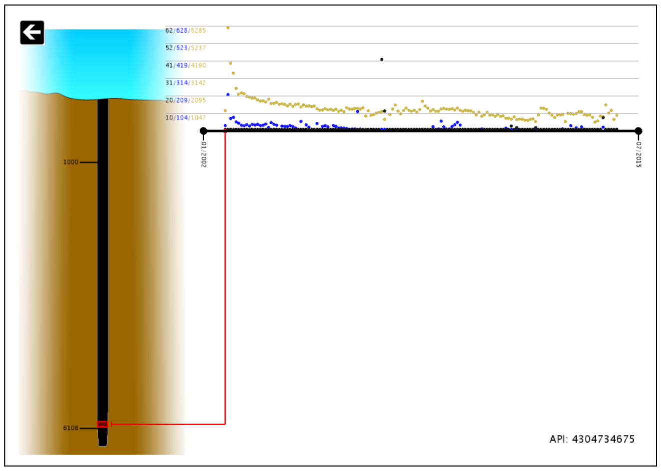

Fig. 3. The second visualization, used for an individual well, presents a schematic of the well, along with an event timeline. This particular well is producing gas in a "normal" pattern.

by reliably predicting how much oil or gas will be accessible.

The variety of data available in this domain is astounding. It ranges from the large-scale seismic imaging, to medium-scale core samples and borehole acoustics, to small-scale micro-CTs and mass spectrometry. There are a variety of drilling techniques used, geologic formations, historical information, and production logs. Our initial studies have focused on data from the United States Geological Survey (USGS) Core Research Center [8], the Arkansas Oil and Gas Commission (AOGC) [5], and the Utah Division of Oil, Gas, and Mining (DOGM) [9]. The case presented here, focuses on a number of databases from DOGM.

Initially, 19 databases, totaling approximately 500 MB are loaded into the indexer. Data is available in near-real-time, as the index updates while it processes. After about 20-30 minutes of indexing, the entirety of the data is available.

The visualization consists of two views. The first is a map view constructed using Unfolding Maps [7], representing the geolocation of a set of wells. The geolocation is queried from a microservice named *geolocation*, which appends each record with a latitude and longitude, if one exists. To reduce the number of wells displayed, a search term can be added, such as *utah mancos shale*[2], returning what is seen in Figure 2. Here, each dot represents a well. Clicking on a dot brings up a detailed view of the well data, as seen in Figure 3.

[2]Mancos shale is a type of shale of geological interest to our collaborators.

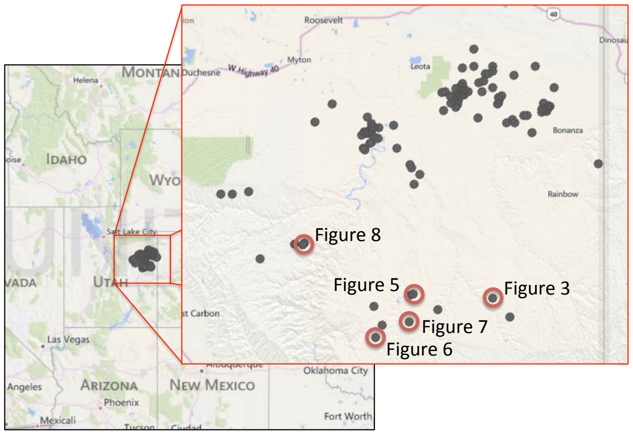

Fig. 4. A map view highlighting the geolocation of the wells used in the case study.

The individual well visualization (Figure 3) is more elaborate, but it emphasizes some of the data features more interesting to our collaborators. At a simple level, our collaborators are interested in correlating the types of hydraulic fracturing treatments used to the oil and gas produced by the well. Of course, many variables impact the type of hydraulic fracturing used. On the left, an illustration of the well can be seen. Within that illustration, a small red region can be seen, indicating a hydraulic fracturing treatment occurred in the well at that depth. This data is queried through a *treatment* microservice that returns a variety of information, including a depth range and date of treatment. That treatment depth is then connected via a red line to a timeline in the top right of the display.

Within the timeline, production data can be seen as well. The *production* microservice returns a series of records with the quantity, date, and type of production. In this case, each dot indicates one month's production. The colors indicate oil, gas, or water in black, yellow, and blue, respectively. The scale of the chart can be seen on the left, with oil, then water, and then gas, each differing by one order of magnitude, respectively.

All of this data is accessed by parsing a variety of schemas, which are queried using the well's API number, a unique identifier assigned to individual wells. This intelligence is built into the microservices.

Using this visualization, we were able to identify a number of types of production wells (described below) in the data. Those well locations are marked in Figure 4.

Normal Production. The first type of well, as seen in Figure 3, is a normal producing well. In this case, the well is mostly producing gas. Experts expect that when a well is first tapped or treated, it will initially produce its largest amount of oil or gas with an algebraic decline over time[3].

Reduced Lifecycle. The algebraic decline in production means that some wells quickly become economically inviable to continue to operate. Once determined, these wells are capped. Figure 5 is one such well.

Erratic Production. The next type of well, as seen in Figure 6, is one with an erratic production cycle. This is interesting because there is a desire to know why such production irregularities occur. The causes could include equipment issues, changes in recovery procedures, material properties of the well substrate, interconnectedness with other wells, etc. Such a well needs further investigation.

[3]The production rate varies typically by $t^{-1/2}$

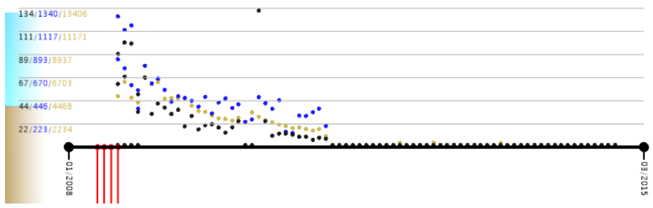

Fig. 5. A well that produces after hydraulic fracturing treatments. However, the well has a short production lifecycle.

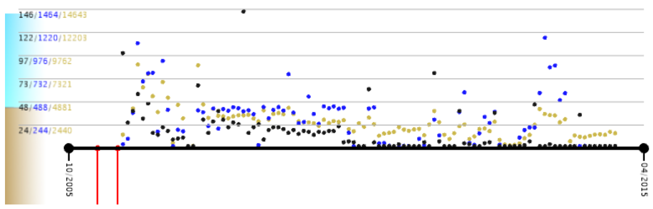

Fig. 6. A well that has hydraulic fracturing treatments applied, yet the well produces at an erratic rate. Such a well would require additional investigation.

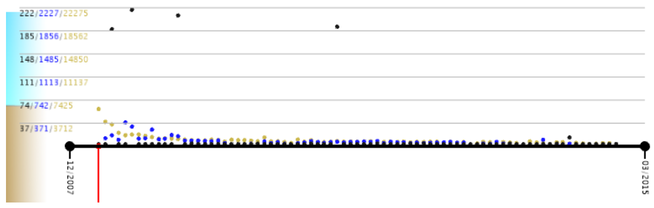

Fig. 7. This well has hydraulic fracturing treatments applied. However, the treatments was unsuccessful, leading to limited production.

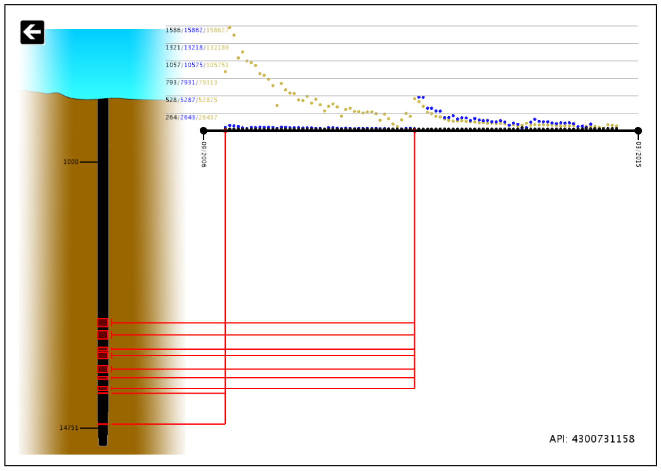

Fig. 8. A well that produced normally after an initial treatment. Later, the production was boosted by receiving a new fracking treatment at a different depth.

Unsuccessful Treatment. In a worst case, some hydraulic fracture treatments have no impact on the well production. In Figure 7, a treatment occurs. However, the well production shows no effect.

Multiple Treatments. In many wells, as production begins to fall, additional hydraulic fracturing procedures can be used to boost the well's production. In the case of Figure 8, an initial fracture in late 2006 led to high gas production. A second set of fractures in 2010 at different depths helped to boost gas production from the well.

6 Discussion & Conclusion

The Klareco approach seeks to revolutionize the visual analysis process by providing the data analyst with quick access to data from their native sources by eliminating the need to perform costly ETL processing before any exploration or analysis.

From a system perspective, Klareco can be quickly deployed on local or remote systems. The system naturally supports both structured and unstructured data, including databases, spreadsheets, images, PDFs, etc. It can integrate both legacy and new data, removing the burden of reformatting or repivoting data into a common schema. It removes the need for a data warehouse to store data after it has been ETLed—only the index needs storing. The index is dynamically updated, such that changes to data sources become quickly visible in the visualization.

From the user perspective, the advantage of such an architecture is speed to exploration. Data are added to the index, and within a few moments, the user is able to start exploring and manipulating their data.

As this architecture is still in its infancy, there remain a number of undirected problems.

Data Pivoting The burden of pivoting data no longer lies in the hands of the ETL engineer. Instead, the visualization and data analyst are responsible for data pivoting. This may require new visualization techniques that optimize such tasks.

Mixed-schema Visualization Not only are the visualizations now responsible for pivoting the data, they are also responsible for visualizing data with mixed schema. The index server may return a huge list of records all with different schema. It is currently the visualization or a new microservice's responsibility to parse all of those schema for data of interest.

Missing Data Visualization From the perspective of a visualization engineer, one of the biggest advantages of ETL is that the data has been cleaned—that is there are no missing or invalid data. The Klareco architecture removes that assumption from the visualization. Now, data arrives at the visualization in nearly unmodified condition.

Security and Data Availability A number of network and security issues created by this approach remain unresolved. These include resiliency issues like dealing with index or microservice unavailability or load-balancing in a cloud environment. Of potentially larger concern are security protections. There are a huge variety of security scenarios one could envision. These deal with access of the original native data sources, to access to the index, to access to specific microservices.

Despite the unanswered nature of these questions, we see them all as relatively tractable problems. Ultimately, the benefits of this approach lie heavily towards the ease of use for the user looking to quickly explore their data by removing or delaying the costly ETL process.

Acknowledgments

We thank our funding sponsor NSF DIBBs ACI-1443046.

References

[1] Apache lucene: http://lucene.apache.org/.
[2] Apache solr: https://lucene.apache.org/solr/.
[3] Apache tika: http://tika.apache.org/.
[4] Apache tomcat: http://tomcat.apache.org/.
[5] Arkansas oil and gas commission: http://www.aogc.state.ar.us/.
[6] Processing: http://www.processing.org.
[7] Unfolding maps: http://unfoldingmaps.org/.
[8] United states geological survey: http://geology.cr.usgs.gov/crc/.
[9] Utah division of oil, gas, and mining: http://oilgas.ogm.utah.gov/.