High-Dimensional Scientific Data Exploration via Cinema

Jonathan Woodring, James P. Ahrens, John Patchett, Cameron Tauxe, and David H. Rogers

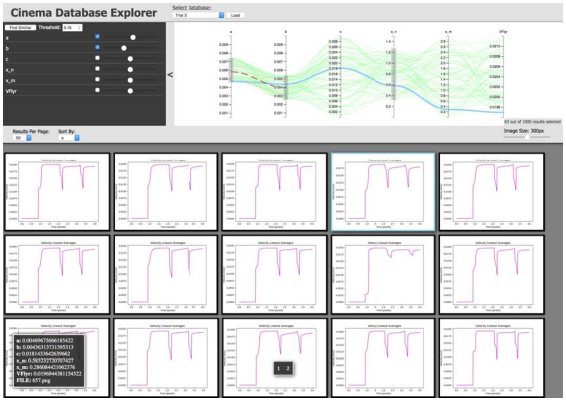

Fig. 1. A *parallel coordinates-based* viewer that allows users to interactively explore large-scale scientific data sets through imagery, by brushing-and-linking across a high-dimensional parameter space. This implementation is a *Cinema viewer* that visualizes the new *Dietrich* specification. *Cinema Dietrich* describes scientific data as relational data sets (*aka* structural data, *aka* tables) with a simple, easy-to-use Comma Separated Values (CSV) format. It is extensibile via additional information (schema meta-data) in JavaScript Object Notation (JSON) for describing complex *Cinema* use cases.

Abstract—Large-scale scientific simulations and experiments generate enormous volumes of data. Data analytics may become a bottleneck to scientific discovery without scalable tools for interactive exploration. *Cinema* was developed as a way to overcome hurdles by providing an exploratory, image database approach for analyzing large scientific data sets. In the following, we present several new methods for *Cinema*: 1) a structured data model that lends itself to querying and database support, 2) support for arbitrary data products beyond images, and 3) parameter exploration through high-dimensional visualization. These changes enrich the types of exporatory visualizations and discoveries that are naturally supported by *Cinema*-style analyses, further enabling data-driven science.

Index Terms—Cinema, high-dimensional data, databases, structural data, parallel coordinates, in situ, parameter exploration, image databases, interactive exploration, big data, exascale supercomputing

1 Introduction

Cinema [3] is a scalable approach for providing interactive, exploratory visualization and analysis for extreme-scale scientific data sets. In the following, we describe several new methods, extending *Cinema*-based analytics: 1) a structured data model that lends itself to querying and database support, 2) support for arbitrary data products beyond images, and 3) parameter exploration through high-dimensional visualization.

Extreme-scale simulations and experiments [1, 2, 4, 10, 19] are leading the charge towards high-fidelity, data-driven scientific discovery. Analysts are collecting, modeling, and visualizing large-scale data sets to generate new insights. This entirely depends on having scalable data analytics, otherwise, discoveries will be severely bottlenecked by the rate that analysts can gain insights from their data.

Due to I/O constraints, large-scale simulations are increasingly moving towards *in situ* analytics to deal with limitations in data movement [5, 6]. While *in situ* methods are able to deal with the CPU to I/O impedance mismatch, it limits the variety of exploratory analysis that can be done during post-processing visualization. The types and number of data products are usually decided upon in advance of a simulation run, which can lead to biases in analysis [13].

Cinema was developed as a way to mitigate this limitation by introducing exploratory analysis to an *in situ* workflow. It achieves this by generating a *database of images* via rendering *in situ* visualizations of a simulation. The image data set samples the *high-dimensional parameter space* of visualization operations, such as camera position or isosurface values. Through sampling the visualization operation parameter space for an exascale data set (order of 10^{18} values), it can be reduced to gigascale or terascale (10^9 or 10^{12} scale data, respectively), and still provide high-impact interactivity: *a large-scale simulation run will generate 1,000s of megascale images that can be explored in post-processing.*

In the research, development, and production use of *Cinema*, we observe several scenarios that lead to new capabilities that capture more scientific data analysis use cases. We describe these new extensions to *Cinema*, its data description, and example viewer implementations with scientific data sets.

2 Related Work

Cinema is a novel framework for visualizing large-scale scientific data sets, as described by Ahrens et al. [3]. Like Tikhonova et al. [23], it uses image-based techniques to scale down massive data sets to interactive, explorable data sets. Recently, Larsen et al. [11] modeled the costs for *in situ* generation of imagery, which is important for measuring the performance of *Cinema* and image-based analysis workflows.

Our new *Cinema*-based techniques utilize database and query-style visualizations for scientific datasets. In particular, visual analytics and information visualization have a long, rich history of database

- *J. Woodring, J. P. Ahrens, J. Patchett, and D. H. Rogers are with the Los Alamos National Laboratory. E-mail: woodring@lanl.gov, ahrens@lanl.gov, patchett@lanl.gov, and dhr@lanl.gov.*
- *C. Tauxe is with the Los Alamos National Laboratory and the New Mexico State University. Email: camerontauxe@lanl.gov.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

and structured data analytics [7, 26]. For scientific data, there have been several different methods for bringing databases into visualization and analysis. Notable is SciDB [16], where Stonebraker et al. have developed databases geared specifically towards scientific analysis workflows. Raddick et al. [21] describe the usage and hosting of the Sloan Digital Sky Survey data on the web and scientific analysis performed with database technology.

In the database model for scientific visualization, there is a need for fast-query and indexing capabilities, due to the scale of datasets and the ubiquity of scalar, floating-point data. In particular, bitmap indexing has been a focal point of research for querying scientific data. Gosink et al. [9] describe indexing HDF5, a structured data file format for scientific data, with bitmap indexing methods. Similarly, Reubel [20] et al. describe query-based visualization and analysis methods using bitmap indexing technology on scientific data.

With databases and query functionality, visualization and analysis systems can be built to utilize these capabilities. Planthaber et al. describe *EarthDB* [14], a scalable earth visualization system built on top of SciDB. Su et al. [22] describe doing correlation and statistical analytics on scientific data sets indexed by bitmap indexing. Databases are able to support high-dimensional analytics and parameter studies, a focal point for the new capabilities in *Cinema* moving to a relational and database data model. In this area, ensemble and high-dimensional analytics in scientific data have been studied by Potter et. al with EnsembleVis [15] and Crossno et al. with Slycat [8].

3 Previous Cinema Work

Cinema is designed as a data description and format for interactive visualization of large-scale scientific data. Focusing on data specification rather than one software implementation or API, *Cinema* provides the ability to deliver many implementations: production quality software and simultaneously allowing research software to continue. This is because a data format specification allows various software implementations to have differing requirements, such as reliability (production) vs. capability (research). For example, an *in situ ParaView Catalyst* adapter (production software) can write a data set that the web-based *Cinema* viewer (research and development software) can display.

In *Cinema*, we use Golden Age era actors as code names for data specification documents. The first two data formats are documented in the *Cinema Astaire* [17] and *Cinema Chaplin* [18] Specifications, found at **http://cinemascience.org/index.php/getting-started/** and summarized in the following.

3.1 A Collection of Rendered Images

The *Cinema* Specification A, *Astaire*, was the first released specification that described the storage format for: 1) a JSON meta-data file, 2) parameter values in the JSON, 3) a string format in the JSON, for converting parameter values into image filenames, and 4) a collection of images on disk. Parameters describe and enumerate a set of dimensions and values for those dimensions. Parameter examples include visualization operator dimensions: camera positions, isosurface values, slicing parameters, color maps, etc., as well as simulation parameters: time step and output variables. Parameter values are converted to an image filename list by taking the Cartesian product of those dimensions and converting them to a string with the format, creating a list of images. That is, an *in situ* workload for creating an *Astaire* data set will generate all possible combinations of parameter values to create output a set of rendered images stored in PNG, JPEG, etc. on the file system.

```
{
  "type": "simple", "version": 1.1,
  "metadata": { "type": "parametric-image-stack" },
  "name_pattern": "{phi}_{theta}.png",
  "arguments": {
    "phi": { "default": 0.0, "label": "phi (degrees)", "type": "range",
      "values": [-90.0, 0.0, 90.0] },
    "theta": { "default": 0.0, "label": "theta (degrees)", "type": "range",
      "values": [-180, 0.0, 180.0] }
  }
}
```

Fig. 2. JSON meta-data for a simple *Cinema* Specification A (*Astaire*) database. It has two parameters, *phi* and *theta*, which describe PNG images on disk with filename string format *"{phi}_{theta}.png"*.

To describe the list of collected images in *Astaire*, parameters, parameter values, and the string format are recorded in a JavaScript Object Notation (JSON) file. In Figure 2, we show the JSON for a common use case in *Cinema*, describing the *phi* and *theta* camera dimensions (parameters) for an orbiting camera. These parameters would be defined with their valid values, like in our example, *-90, 0, and 90* (degrees) for *phi* and *-180, 0, and -180* (degrees) for *theta*. Given the format string, "{phi}_{theta}.png" (Python-style string formatting), output images are described by taking the Cartesian product of *phi* and *theta*, and generating image filenames with the value pairs using the format string, some examples being: *-90_-180.png*, *0_0.png*, *90_180.png*, etc.

3.2 A Collection of Compositible Images

Cinema Specification C, *Chaplin*, was the second public release of a *Cinema* specification, providing additions to *Astaire*. The first addition describes floating point ("raw") images and compositible image layers. With this, scientific data sets may be rendered without a color transfer function and saved as intensity or floating point images with z (depth) buffers. This allows for both interactive coloring and layered rendering in post-processing by *Cinema* viewers. It reduces the number potential number of images that need to be saved during *in situ* rendering, saving computation time, by moving the coloring and compositing into the viewer process. *Chaplin* also provides the capability for describing a *sparse* sampling of the parameter space, limiting the collection of images. It does so by describing which parameter values are "valid" conditionally upon other parameter values. For example, an *isosurface* parameter value of 3.0 might only be valid when the rendered *variable* parameter is "temperature."

4 A Database of Scientific Data Products

The newest *Cinema* Specification, D for *Dietrich* [24], uses a structural data model, i.e., a relational table, to represent parameters and data products: 1) a simple CSV database for describing data products and parameters, 2) a collection of data products on disk indexed by the CSV, and 3) an optional JSON file for describing advanced use cases. This is different from previous specifications where output images were implicitly indexed by string formatting parameter values into image filenames. *Dietrich* completely supports the *Astaire* and *Chaplin* use cases, where image filenames are represented by a column in a table. Parameters are related data, where a row represents an relationship between parameter values and image filenames.

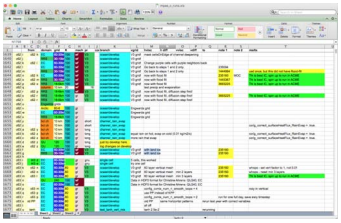

Fig. 3. An example of a working scientist's spreadsheet in Excel. Scientists keep a record of their parameters and data products, which is similar to a *Cinema* database.

There are several reasons for moving to an explicit, relational data (structural, table) model in *Dietrich*:

1. **Scientific spreadsheets and parameter exploration** – Scientists already use spreadsheets in their workflow, and it is quite common for them to use them to record their work and data products, like in Figure 3. With *Dietrich*-based *Cinema* viewers, we are able to directly visualize these data sets that scientists already have in their possession. Many scientists perform parameter studies that consist of non-uniformly sampled data in a parameter space, which are described in these spreadsheets.
2. **Comma Separated Values (CSV) file format** – Structured data can be represented by CSV that is a more universal data format than JSON. CSV is easier to understand and implement by readers,

writers, and end users, providing a fast-path to adoption and on-ramp to *Cinema*. Furthermore, it is likely that a scientist already uses CSV (or CSV-like) data sets in their workflow.

3. **Arbitrary image file paths** – *Astaire* and *Chaplin* were designed around generating databases of images and viewers for exploring those images. Image filenames were restricted by a string format, mapping parameter values to filename paths. In *Dietrich*, this is relaxed as a filename can be any POSIX pathname or URI, as filenames are columns in the table, allowing a broader range of file storage options and locations.
4. **Arbitrary data types and products** – Like with arbitrary filenames, *Dietrich* is able to easily support visualizing arbitrary data types other than images. For example, *Astaire* and *Chaplin* were restricted to visualizing images. To visualize plots, it required pre-rendering them as images. In *Dietrich*, the data used to create the plots can be directly represented as column data in the CSV. This allows the data to be interactively plotted, rather than requiring pre-rendered images.
5. **Output to input queries** – *Astaire* and *Chaplin* treat parameters as *input* values that are mapped into *output* images. Therefore, exploration is generally performed by searching the input parameter space to find the corresponding output images. To do *output to input* exploration (searches on image data to parameter values), it requires parsing the image filename based on the image filename string formatting. *Dietrich* is able to support these types of "output to input" explorations in a natural way, due to its relational data model, because "input" or "output" are just labels on columns. As such, any dimension (parameter) can be part of a query and return results for any other dimension, regardless if it is an input or output.
6. **Relational semantics and databases** – Related to the previous point, it is easy to represent any number of columns as the result of a query (i.e., joins and projections). Also, a rich set of relational data semantics and *database support* can be brought to bear, such as indexing and query processing. For example in our *Dietrich* query-based viewer, *Cinema* CSV data is internally represented by a *SQLite* database, providing the majority of SQL functionality.
7. **Sparse and biased sampling of parameter spaces** – In *Astaire*, the visualization operator space is sampled by the Cartesian product of parameter values to generate the list of images. *Chaplin* provides sparse sampling in a limited way, by describing a set of "constraints" per dimension that have to be programmatically tested and does not easily support random sampling. In *Dietrich* due to its relational model, samples can be arbitrary points in a high-dimensional space (parameter values to data product relationships) due to explicit row data. This provides the capability to represent a sparse, non-uniform sampling of any parameter space, supporting visualization of scientific parameter studies, like in Figures 1, 3, and 6.

In particular with *Astaire* and *Chaplin*, all "useful" information is only contained in the images. This is due to the parameter sampling process, where the Cartesian product of dimensions will decorrelate parameter values and the information *across* dimensions. With *Dietrich*, we can describe parameter correlations through a biased sampling of a high-dimensional parameter space. This assumes that the parameter space is purposely sampled (i.e., during the *in situ* data generation step) in a data-driven manner, such as importance sampling [12].

4.1 Simple Database Format

In previous versions of the *Cinema* Specifications, the parameter values and string format meta-data are described in a JSON file. With *Dietrich*, we use Comma Separated Values (CSV) to describe this same set of meta-data as a table. Each parameter is a column in the CSV, including the image filenames, such that each row describes the relationship between parameters and image filenames. Figure 4 shows an example of *Dietrich* CSV file, which describes the same database as Figure 2. We use CSV for *Cinema Dietrich*, rather than picking a database implementation, owing to our previous focus on both specification and format instead of software implementation. Also, there is the added difficulty of running database servers in a supercomputing/high-performance computing (HPC) environment, where many of our end users of *Cinema* will compute their data. Additionally, they may already have CSV files or other tabular data (such as spreadsheets) that can be easily converted to CSV and read by *Cinema*. Finally, CSV is easy to read, write, share with other tools, and ingest into database backends for viewer implementations (independent of the data format).

phi,	*theta*,	*FILE*
-90,	-180,	-90_-180.png
-90,	0,	-90_0.png
-90,	180,	-90_180.png
0,	-180,	0_-180.png
0,	0,	0_0.png
0,	180,	0_180.png
90,	-180,	90_-180.png
90,	0,	90_0.png
90,	180,	90_180.png

Fig. 4. A *Cinema* Speficiation D, *Dietrich*, database stored in CSV format. This is the same data set as Figure 2, showing parameter value to image filename relationships as explicit rows.

Our design focus for *Cinema* is on ease-of-use, like our previous specifications and currently, we are not concerned about the scalability of CSV. We expect much of "heavy-weight" data products, such as images, are stored in external files on the file system. In the current specification, we denote that a column represents external file data, like images, with a special header label (i.e., *FILE*) and the file type is determined by filename extension. This is similar to our earlier *Cinema* JSON format, where the CSV acts as a way to index high-dimensional parameter values to data products. Some data, like data columns intended to be shown in plots, will be stored in the CSV for visual interactivity. In the future if scalability becomes an issue, we can move to a different implementation updating the specification.

4.1.1 Optional Schema for Complex Use Cases

We expect that the CSV will cover the majority of *Cinema* use cases, allowing for easy initial adoption and visualization of scientific data sets. A positive and negative of CSV is its simplicity. *Dietrich* extends it to support files, through the *FILE* keyword in the CSV header. The simplicity makes it difficult to describe advanced *Cinema* use cases and data types. This is important to describe complex data columns to viewers because an implementation might want to create specialized interfaces, such as a trackball interface if it understands that a column represents camera data.

```
"temperature-column": {
  "type": "scalar", "io": "output", "label": "temperature",
  "arguments": {
    "value": "temp", "range": [-10.0, 30.0],
    "interpolate": "linear", "units": "Celcius" }
}
```

Fig. 5. A portion of a *Dietrich* JSON meta-data for *scalar* columns.

For these advanced use cases, we provide a path to extend the CSV stored in an additional, *optional* meta-data JSON file. In *Chaplin*, we have specialized JSON keys that describe *Cinema* use cases, like cameras, image layers, and raw image formats. In *Dietrich*, we provide a specification describing these data, a flexible JSON meta-data format for *semantic information* or a *schema* on top of the CSV. A snippet of a JSON meta-data for describing continous, scalar data is shown in Figure 5. JSON has multiple benefits as a meta-data descriptor to the CSV: it is extensible by the way of adding new key-value pairs to the specification, has many available parsers for various languages, and JSON Schema [25] provides a way to validate the meta-data.

This is in contrast to extending the set of special keywords for CSV column headers (like *FILE*). The main reason for not using additional header keywords is that they are insufficient to capture and communicate complex use cases, like image layers in *Chaplin*. In a simple use case, if a parameter (column) is an *input* that implies viewers ought have a control, like a slider. While *outputs* imply columns should be the visual result of manipulating controls and/or querying the CSV. Thus, there is a need to communicate intent beyond simple machine types.

4.2 Viewers (User Interaction and Exploration)

Given our new relational data model for *Cinema Dietrich*, we will discuss two viewers that were created around these data models: a parallel coordinates-style viewer and a query-based viewer.

4.2.1 Parallel Coordinates Viewer

In *Astaire* and *Chaplin*, the visualization parameter space was fully enumerated, as the Cartesian product over the dimensions, creating a *dense* sampling of the parameter space. This meant that viewers primarily create user interfaces using sliders and/or drop-downs, because any possible configuration would result in a valid parameter combination and output image selection. This was intended, as most of the initial *Cinema* use cases were built around densely sampling a camera parameter space around a visualized object to provide smooth, visual interactivity via pre-rendered images.

By moving to a relational data model in *Dietrich*, this opened up the possibility of *sparse* sampling of parameter spaces. This creates a user interface design problem for parameter selection and image exploration. We were unable to easily use sliders as a way to explore the parameter space, as with *Astaire* and *Chaplin*. A combination of sliders, in a sparse representation, may not select a valid row in the database. To solve this problem, we initially considered a user interface where the sliders would snap to the "nearest valid row" in the database, i.e., nearest neighbor search in one dimension, upon moving one of the sliders. Sliders could get locked into a local-minima of the parameter space, and it is unintuitive, as the other sliders would be moving as the user manipulated one of the dimensions (sliders).

To solve this problem, we developed a *parallel coordinates* explorer as a way to interact and select rows in a *Cinema* database, as shown in Figure 1. This solved our user interface problem, as now end users can select individual rows by selecting a point (line segment) or ranges of rows by brushing on an axis (dimension). This essentially creates pairs of slider widgets (min-max range pairs for each parameter by brushing on a dimension) with benefits of parallel coordinate visualization:

1. **Parameter space visualization** – With the capability to sparsely and importance sample the parameter space, we can provide information about the values and correlations between dimensions in the high-dimensional space. Parallel coordinates provide a way to see all of the dimensions, database rows, and parameter relationships to outputs, simultaneously. In particular, the data that are shown in the previous figures are from scientific parameter studies where the parameter space is non-uniformly sampled, with multiple inputs and outputs.
2. **Brushing and linking** – Range queries on dimensions are applied through brushing and linking on each dimension, updating the selected rows and output images immediately. Future *Cinema* viewer implementations may include a *scatterplot matrix* or interleaved scatterplots between parallel coordinate axes, allowing users to see bi-variate relationships that are easier to see in 2D, like non-linear relationships.

4.2.2 Query-based Viewer

By having a structured data model, we can directly apply relational-style queries (*SQL* in our implementation) on the *Cinema* parameter space to explore images and data, as shown in Figure 6 with a six-dimensional data set. This is similar to the visual range queries that we are able to perform via the previous parallel coordinates viewer, but instead exploration is performed with *SELECT* and *WHERE* statements. To execute the queries, we used *SQLite* to internally back the CSV data and parse SQL code.

For example, we can easily load two separate *Dietrich* databases, combine them together with a join on a common parameter, and saving the result as a new view. We used this workflow on the data in Figure 6 to combine two *Cinema* data sets into one explorable database. In our example, one data set has image data indexed by *time*, while the other has scalar data with *time*, *tracer*, *absorption*, *scattering*, and *multigroup*. Combining them by an *inner join* on *time*, we can show

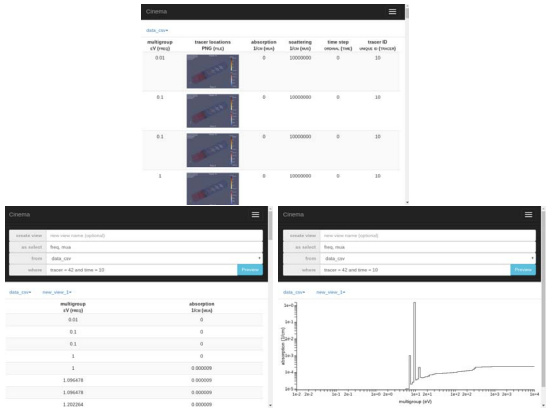

Fig. 6. Examples of performing exploration with the *query-based Cinema* viewer. The top image shows all of the data in the database, including images. In the bottom-left image, we show the result of the query on the data set that does not result in imagery. In the bottom-right image, we use the JSON meta-data to describe that the same query can be plotted.

the corresponding images data with the scalar data, as shown in the top image of Figure 6.

In the bottom-left image of Figure 6, we show the ease of selecting and visualizing two parameters (dimensions) based on queries using two of the other parameters. In this case, we select *multigroup* and *absorption*, such that *time* equals 10 and *tracer* equals 42. By default, the viewer implementation will show query results as a table, without additional semantic knowledge, but we can do more. Utilizing the JSON meta-data for describing columns in the CSV, we annotate that *multigroup* and *absorption* are continuous scalar data. With this knowledge, the viewer can understand that the query is returning two columns of scalar data, and therefore, can line plot it. The bottom-right image of Figure 6 shows the result, where the viewer is using the semantic knowledge provided by the JSON meta-data to line plot the query result instead of showing it as a table.

5 Discussion on Machine Learning in Cinema

Currently, we have several papers in preparation that study the use of machine learning and statistical learning techniques for *Cinema* databases. What is important about these unpublished works, in general, is that automated data modeling techniques augment and accelerate human-in-the loop data exploration. With *Cinema's* relational data model, we can easily augment an existing scientific database by writing machine learning algorithms as unattended *Cinema* readers and writers. For example, the areas that we are currently exploring is using machine learning to continuously apply "feature finding" (classification) methods to augment and update an existing *Cinema* database over time.

Essentially, machine and statistical learning allows *Cinema* datasets to become "living" and "growing," as they stop being static through automatic computer data modelling. With our relational data model and continually running learning algorithms, viewers will automatically pick up any new parameters (columns) that have been added to a *Cinema* database, providing new user interface controls. In essence, automatic data modelling can provide "dimensionality reduction", reducing high-dimensional data to fewer dimensions. *Cinema* can then automatically visualize these new parameters (dimensions) through sliders, if reduced to 1D, or using our new parallel coordinates viewer.

6 Conclusion

We have demonstrated the next advances in *Cinema*-style scientific visualization and analysis utilizing a new relational data model. This provides many benefits: ease of high-dimensional parameter description and exploration, supporting high-dimensional visualization like parallel coordinates, database and query support, and the ability to visualize and analyze many existing scientific data sets.

REFERENCES

[1] S. Ahern, A. Shoshani, and K.-L. Ma. Scientific Discovery at the Exascale. Report from the DOE ASCR 2011 Workshop on Exascale Data Management, Analysis, and Visualization, DOE Office of Science ASCR, Houston, TX, Sept. 2011.

[2] J. Ahrens, B. Hendrickson, G. Long, S. Miller, R. Ross, and D. Williams. Data Intensive Science in the Department of Energy. Technical Report LA-UR-10-07088, Los Alamos National Laboratory, Oct. 2010.

[3] J. Ahrens, S. Jourdain, P. OLeary, J. Patchett, D. H. Rogers, and M. Petersen. An Image-Based Approach to Extreme Scale in Situ Visualization and Analysis. In *SC14: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 424–434, Nov. 2014.

[4] J. Ahrens, D. Rogers, and B. Springmeyer. Visualization and Data Analysis at the Exascale. White Paper for the National Nuclear Security Administration (NNSA) Accelerated Strategic Computing (ASC) Exascale Environment Planning Process LLNL-TR-474731, DOE NNSA ASC, 2011.

[5] U. Ayachit, A. Bauer, E. P. N. Duque, G. Eisenhauer, N. Ferrier, J. Gu, K. E. Jansen, B. Loring, Z. Lukic, S. Menon, D. Morozov, P. O'Leary, R. Ranjan, M. Rasquin, C. P. Stone, V. Vishwanath, G. H. Weber, B. Whitlock, M. Wolf, K. J. Wu, and E. W. Bethel. Performance analysis, design considerations, and applications of extreme-scale in situ infrastructures. In *SC16: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 921–932, Nov 2016.

[6] A. C. Bauer, H. Abbasi, J. Ahrens, H. Childs, B. Geveci, S. Klasky, K. Moreland, P. O'Leary, V. Vishwanath, B. Whitlock, and E. W. Bethel. In situ methods, infrastructures, and applications on high performance computing platforms. *Computer Graphics Forum*, 35(3):577–597, 2016.

[7] C. Chabot. Demystifying Visual Analytics. *IEEE Computer Graphics and Applications*, 29(2):84–87, Mar. 2009.

[8] P. J. Crossno, T. M. Shead, M. A. Sielicki, W. L. Hunt, S. Martin, and M.-Y. Hsieh. Slycat Ensemble Analysis of Electrical Circuit Simulations. In *Topological and Statistical Methods for Complex Data*, Mathematics and Visualization, pages 279–294. Springer, Berlin, Heidelberg, 2015. DOI: 10.1007/978-3-662-44900-4_16.

[9] L. Gosink, J. Shalf, K. Stockinger, K. Wu, and W. Bethel. HDF5-FastQuery: Accelerating Complex Queries on HDF Datasets using Fast Bitmap Indices. In *18th International Conference on Scientific and Statistical Database Management (SSDBM'06)*, pages 149–158, 2006.

[10] C. Johnson and R. Ross. Visualization and Knowledge Discovery: Report from the DOE/ASCR Workshop on Visual Analysis and Data Exploration at Extreme Scale. Technical report, Department of Energy Office of Science ASCR, Oct. 2007.

[11] M. Larsen, C. Harrison, J. Kress, D. Pugmire, J. S. Meredith, and H. Childs. Performance Modeling of in Situ Rendering. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '16, pages 24:1–24:12, Piscataway, NJ, USA, 2016. IEEE Press.

[12] B. Nouanesengsy, J. Woodring, J. Patchett, K. Myers, and J. Ahrens. ADR visualization: A generalized framework for ranking large-scale scientific data using Analysis-Driven Refinement. In *2014 IEEE 4th Symposium on Large Data Analysis and Visualization (LDAV)*, pages 43–50, Nov. 2014.

[13] J. M. Patchett, B. Nouanesengsy, G. Gisler, J. Ahrens, and H. Hagen. In Situ and Post Processing Workflows for Asteroid Ablation Studies. 2017.

[14] G. Planthaber, M. Stonebraker, and J. Frew. EarthDB: Scalable Analysis of MODIS Data Using SciDB. In *Proceedings of the 1st ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data*, BigSpatial '12, pages 11–19, New York, NY, USA, 2012. ACM.

[15] K. Potter, A. Wilson, P. T. Bremer, D. Williams, C. Doutriaux, V. Pascucci, and C. R. Johnson. Ensemble-Vis: A Framework for the Statistical Visualization of Ensemble Data. In *2009 IEEE International Conference on Data Mining Workshops*, pages 233–240, Dec. 2009.

[16] M. J. Raddick, A. R. Thakar, A. S. Szalay, and R. D. C. Santos. Ten Years of SkyServer I: Tracking Web and SQL e-Science Usage. *Computing in Science Engineering*, 16(4):22–31, July 2014.

[17] D. Rogers, D. DeMarle, J. Ahrens, and J. Patchett. Cinema Simple Database Specification v1.1. Technical Report LA-UR-15-20572, Los Alamos National Laboratory, 2014.

[18] D. Rogers, J. Woodring, J. Patchett, D. DeMarle, and B. Geveci. Cinema Database Specification Chaplin Release v1.0. Technical Report LA-UR-17-20645, Los Alamos National Laboratory, 2017.

[19] R. Rosner and A. S. on Exascale Computing. The Opportunities and Challenges of Exascale Computing. Summary Report of the Advanced Scientific Computing Advisory Committee (ASCAC) Subcommittee, DOE Office of Science ASCR, Nov. 2010.

[20] O. Ruebel, E. W. Bethel, M. Prabhat, and K. Wu. Query-Driven Visualization and Analysis. Technical Report LBNL-6323E, Ernest Orlando Lawrence Berkeley National Laboratory, Berkeley, CA (US), Nov. 2012. DOI: 10.1201/b12985-10.

[21] M. Stonebraker, P. Brown, D. Zhang, and J. Becla. SciDB: A Database Management System for Applications with Complex Analytics. *Computing in Science Engineering*, 15(3):54–62, May 2013.

[22] Y. Su, G. Agrawal, J. Woodring, A. Biswas, and H.-W. Shen. Supporting Correlation Analysis on Scientific Datasets in Parallel and Distributed Settings. In *Proceedings of the 23rd International Symposium on High-performance Parallel and Distributed Computing*, pages 191–202, New York, NY, USA, 2014. ACM.

[23] A. Tikhonova, H. Yu, C. D. Correa, J. H. Chen, and K.-L. Ma. A Preview and Exploratory Technique for Large-scale Scientific Simulations. In *Proceedings of the 11th Eurographics Conference on Parallel Graphics and Visualization*, EGPGV '11, pages 111–120, Aire-la-Ville, Switzerland, Switzerland, 2011. Eurographics Association.

[24] J. Woodring, D. Rogers, J. Ahrens, and J. Patchett. Cinema Database Specification Dietrich Release v1.0. Technical Report LA-UR-17-25072, Los Alamos National Laboratory, 2017.

[25] A. Wright. JSON Schema: A Media Type for Describing JSON Documents. http://json-schema.org/latest/json-schema-core.html, Apr. 2017.

[26] L. Zhang, A. Stoffel, M. Behrisch, S. Mittelstadt, T. Schreck, R. Pompl, S. Weber, H. Last, and D. Keim. Visual analytics for the big data era; A comparative review of state-of-the-art commercial systems. In *2012 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 173–182, Oct. 2012.