Lawrence Berkeley National Laboratory
Recent Work

Title
Web-based visual data exploration for improved radiological source detection

Permalink
https://escholarship.org/uc/item/14z1h9gr

Journal
Concurrency Computation, 29(18)

ISSN
1532-0626

Authors
Weber, GH
Bandstra, MS
Chivers, DH
et al.

Publication Date
2017-09-25

DOI
10.1002/cpe.4203

Peer reviewed

eScholarship.org Powered by the California Digital Library
University of California

Accepted author version, published version available at https://doi.org/10.1002/cpe.4203

Web-based Visual Data Exploration for Improved Radiological Source Detection

Gunther H. Weber[1,2,*], Mark S. Bandstra[1], Daniel H. Chivers[1], Hamdy H. Elgammal[1], Valerie Hendrix[1], John Kua[1], Jonathan S. Maltz[1], Krishna Muriki[1], Yeongshnn Ong[1], Kai Song[1], Michael J. Quinlan[1] Lavanya Ramakrishnan[1], Brian J. Quiter[1]

[1] Lawrence Berkeley National Laboratory, Berkeley CA, 94720

[2] University of California, Davis, CA, 95616

Abstract

Radiation detection can provide a reliable means of detecting radiological material. Such capabilities can help to prevent nuclear and/or radiological attacks, but reliable detection in uncontrolled surroundings requires algorithms that account for environmental background radiation. The Berkeley Data Cloud (BDC) facilitates the development of such methods by providing a framework to capture, store, analyze and share data sets. In the era of big data, both the size and variety of data make it difficult to explore and find data sets of interest and manage the data. Thus, in the context of big data, visualization is critical for checking data consistency and validity, identifying gaps in data coverage, searching for data relevant to an analyst's use cases and choosing input parameters for analysis. Downloading the data and exploring it on an analyst's desktop using traditional tools is no longer feasible due to the size of the data. This paper describes the design and implementation of a visualization system that addresses the problems associated with data exploration within the context of the BDC. The visualization system is based on a JavaScript front-end communicating via REST with a back end web server.

1 Introduction

National security increasingly relies on access to real-time data analyses of possible threats including nuclear or radiological attacks. *Radiation detection* provides the means of detecting, locating and characterizing nuclear or radiological material that is necessary to prevent such an attack. However, the complexity and variability of benign background radiation in an uncontrolled environment can confound detector systems, worsening their sensitivity or causing false alarms. To mitigate this loss of effectiveness, detection algorithms could leverage contextual data to better account for environmental complexities. Immature and/or incomplete data sets hinder the development and testing of such methods and can result in approaches that are tuned to data that represent less diversity and complexity than is encountered in real search activities. The design of general algorithms must be based on a broad set of data sets that capture a wide variety of cases and environments.

The Berkeley Data Cloud (BDC) provides a framework that supports storing and managing data sets collected for the development of such algorithms along with analysis results. The BDC serves to manage data for several projects—including the Radiological Multisensor Analysis Platform (RadMAP) [1] and the Airborne Radiological Enhanced-sensor System (ARES)—that explore the use of various platforms—such as vehicles (RadMAP) or aircraft (ARES)—for developing capabilities to perform radiological searches while leveraging contextual information. Collected data sets consist of baseline data for background radiation for regions across the United States, radiation sensor responses to radiological threat materials and examples

*Corresponding author: GHWeber@lbl.gov

of such materials in complex environments among others. Analyses—such as calibrating and synchronizing measurement results, subtracting background radiation, or "injecting" an artificial source in previously recorded background radiation—provide additional data products that in turn are also added to BDC. In total, BDC currently provides access to many terabytes of raw and derived data, which is expected to grow to approximately 15 TBytes.

The goal of the Berkeley Data Cloud software stack is to provide end-users easy and fast access to the large amounts of data in the system. Central to BDC is a REST-based web service that we refer to as the *data service*. The data service provides access to the data using a wide array of technologies, including a SQL database to store meta data, HDF5 [6] to store measurements in a self-describing format, and FastBit [12] and FastQuery [4] to index scientific data for rapid access. However, to make efficient use of the data service requires that analysts are able to find regions of interest and develop deep understanding of the details of the data. The volume and variety of the data in BDC makes this uniquely challenging for the end users. The visualization tools available in BDC provide a framework that allows the user to explore the data before more detailed analysis and is the focus of this paper. We present the design choices in the BDC architecture that support the visualization system and discuss the components of the system in detail.

The visualization tools in BDC provides four primary functionalities (i) *workspace manager:* identify and validate data sets of interest, (ii) *data browser:* identify regions and parameters of interest in the data sets, (iii) *waterfall plot:* interactive exploration of radiation spectra and (iv) *source inject interface:* set up data for analyses.

The rest of the paper is organized as follows. In Section 2, we provide a brief overview of the BDC architecture that is necessary to understand the visualization tools. Section 3 describes the design and implementation of the visualization system, and Section 4 provides an overview of related work. We summarize lessons learned from building visualization tools for big data defense and homeland security applications in Section 5 and conclude in Section 6.

2 Background: BDC Architecture and Terminology

The BDC's Data Service provides data registration and enables data querying and analyses through both programmatic and web-based interfaces. BDC has four primary architecture components (i) *data ingest*, (ii) *data storage*, (iii) *web-based visualization system*, and (iv) *workflow infrastructure* to support analyses. In this section, we will summarize some of the key concepts of the overall BDC architecture.

2.1 Terminology

The BDC supports a variety of projects that use different systems with one or more sensors to collect data. These systems include vehicles, such as trucks for RadMAP or aircraft for ARES, as well as static sensor setups for calibration and more precise measurements. The individual projects refer to these systems as *platform* on which sensors are mounted to collect data. Each sensor produces a *data stream* of contiguous recorded *data elements*. For example, one sensor may record the current global positioning system (GPS) location as four data streams for longitude, latitude, altitude and time stamps, where each data element in a stream corresponds to one recorded position. Similarly, a gamma-ray detector may record energy and time for detected events as two data streams. Different sensor systems may record with different periodicities.

Data groups combine *data streams* recorded at the same periodicity, i.e., that have the same number of elements and where the n-th element in each stream is recorded at the same time (see Figure 1). Organizing sensor data into Data groups accounts for the fact that (i) there are multiple sensor systems; (ii) sensors in the same system usually record data with the same periodicity and all data streams consist of data elements collected at the same time; and (iii) different sensor systems usually operate independently and record data at varying periodicities. Some data streams record data at regular time intervals, such as GPS and Environment in Figure 1, while others may collect events occurring at unpredictable times, like the Spectrometer in the figure.

Data collections are organized as distinct *runs*, corresponding, e.g., to a single helicopter flight (ARES) or a single pass by an object of interest. During runs, data streams are recorded to files. However, there is not necessarily a one-to-one correspondence between runs and files since interruptions in a single run might result in multiple files per run or multiple runs might be recorded in the same file.

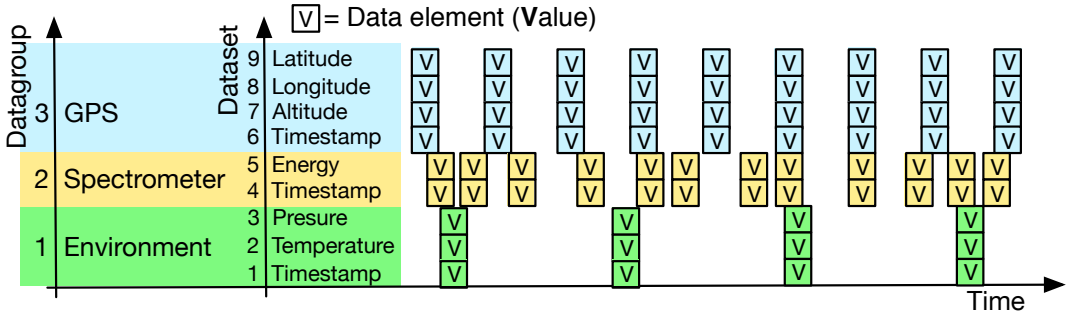


Figure 1: Data groups consist of data streams with values—or data elements—recorded at the same time. This figure shows three data groups: GPS, Spectrometer, and Environment—highlighted blue, yellow and green respectively. Each data group contains multiple data sets: GPS consists of Latitude, Longitude, Altitude, and Timestamp; Spectrometer consists of Energy and Timestamp; and Environment consists of Pressure, Temperature and Timestamp. Within each data group, values/data elements—denoted as boxes containing the letter "V" in the figure—are collected at the same periodicity for all individual streams. We note that the periodicity is not necessarily constant. For example, while GPS and Environment collect data at regular intervals, the timing of events collected by the Spectrometer occur stochastically.

Synchronization correlates data between different data groups using time stamps. To avoid inaccuracies due to clock drift, data streams are usually only correlated per run. In addition to data sets collected by measurement, the system also stores *data products*, which are data sets produced by analysis. These data products can be uploaded by participating vendors/collaboration partners during algorithm development and testing. These data products are treated similar to sensor data in the BDC system.

Sensors, their data sets and data groups, platforms, runs and data sets produced by analyses are all given names within the BDC. These names are assigned and managed within the database, such that absolute paths to data are defined. For example, a list of gamma-ray energies measured by an RSI gamma-sensor is named, 'Energies', and is in the 'SimListMode' data group, which is associated with the 'RSIRadDetector' sensor, which was fielded on 'RSL_HELO' platform. These names are found throughout the data exploration components of the visualization system.

2.2 Berkeley Data Cloud Data Storage Infrastructure

The BDC uses a tiered storage to capture the data and metadata collected during runs (Figure 2). A SQL database (MariaDB) stores meta data about platforms, sensors, runs and files as well as their relationship to each other. While a full description of the database schema is beyond the scope of this paper, the following information is relevant to the visualization aspect of the project:

- A run table stores information about the individual collection runs, including name, start and end time as well as a link to the platform used to collect the data.
- Data sets and data groups are also represented in the database. It is possible to identify all data sets (data streams) belonging to each data group as well as which data set provides time stamps for synchronizing data.
- Information about files is also stored in the database, making it possible to identify files with the data belonging to a particular run

The BDC uses the hiearchical HDF5 data format [6] to store multiple data streams in a single file. HDF5 supports data sets of arbitrary dimensionality as well as efficient access to data subsets. Using HDF5, it is also possible to associate attributes with data sets providing a more detailed description of the data, such as adding information about units etc.

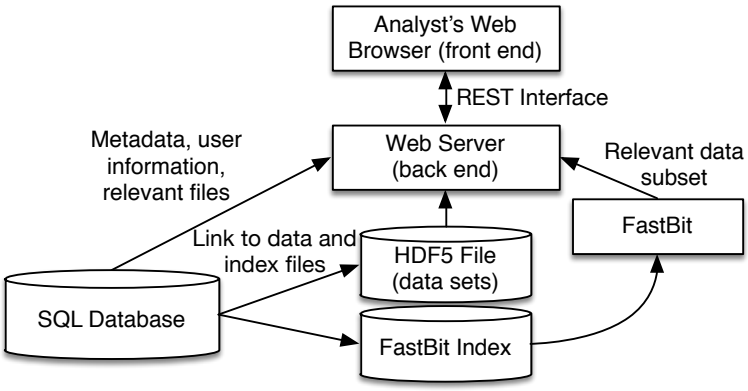


Figure 2: Architecture of the Web-based visualization system for the Berkeley Data Cloud (BDC).

BDC relies on FastBit [12] and FastQuery [4] for efficient data access. A FastBit index for the scientific data provides an efficient means to identify relevant data subsets in terms of range queries. FastBit also enables quick computation of two-dimensional histograms from the index. FastQuery links FastBit to HDF5 by providing means to compute an index for data in HDF5 files and to store the index itself in HDF5 format. The BDC web visualization system uses the SQL database to obtain a list of files (and index files) containing information for a data set for a run. Subsequently, it uses FastQuery and the FastBit index to determine the portion of the file corresponding to a run.

The BDC supports the concept of *queries* to specify data subsets of interest. Queries are a boolean combination of relational criteria, such as $altitude > 500$ AND $altitude < 750$ AND $time > 400$. FastBit/FastQuery support this type of query very efficiently, making it possible to generate filtered files for a user quickly and effectively. Since all data sets are indexed, it is possible to specify range queries on any data set and load, e.g., only data belonging to a given run obtained in a given altitude range. The most common queries select contiguous geographic ranges—i.e., rectangular regions in latitude and longitude—or time ranges.

3 Visualization Design and Implementation

The visualization system in the context of this project serves multiple purposes. The *workspace manager* (Section 3.1) allows analysts to identify data sets of interest based on platforms, sensors, etc. The *data browser* (Section 3.2) allows analysts to examine collected data in detail, e.g., to check region coverage and identify interesting data subsets. The *waterfall plot* (Section 3.3) supports visual inspection of radiation spectra collected during a particular run, e.g., to identify passing an artificial radiation source. Finally, the *source inject interface* (Section 3.4) simplifies specifying parameters for analysis tasks, currently, specifying location and other parameters for a simulated artificial radiation source.

The following requirements guided the design of the visualization system: (i) Users need to access and interact with the large data sets without installing additional software and (ii) BDC has large data sets—several gigabytes per run—making it inconvenient to transfer complete data sets to the user interactively; but meaningful visual exploration requires interactivity. We address these requirements through web-based visual data exploration, enabling the analyst and consumers of analysis products to interact with the data remotely. Our visualization system couples the browser-based front end with a custom web server back end, using a REST protocol. The REST interface is used to request additional data during visual exploration based on user's request. This approach limits the amount of data transferred initially to begin the analyses while also providing full data access during data exploration.

Many analyses require sub-setting the data or computing statistics such as histograms. The data sets collected by the projects are too large making it prohibitive to perform these calculations on-the-fly using

current methods. The need to change parameters—such as histogram bin sizes—prevents pre-computing this information. In our system, we have developed an innovative real-time solution that uses hierarchical HDF5 file format and FastBit indexing to accelerate identification and access of relevant data portions and computation of these statistics. Our approach enables reasonable response times during the analysis.

Web Technologies used in the Visualization System: Our visualization system is implemented in JavaScript and uses a combination of popular web technologies and third-party libraries to facilitate the design of an interactive interface. We use Bootstrap (`http://getbootstrap.com`) that provides a framework to layout responsive web pages for the user interface. We use jQuery (`https://jquery.com`) which is a small, fast JavaScript library that simplifies HTML document traversal and modification, event handling, animation and asynchronous data requests. Our system uses jQuery to request data from the back end as the need arises and to modify the web page to display analysis results and update the user interface based on loaded data. Our system uses D3.js to display histograms and line plots as well as providing basic interaction mechanisms for specifying data selections and zooming into data subsets. Data Driven Documents (D3.js) [3] build on jQuery to provide higher level primitives for manipulating web pages based on data to be visualized. Polymaps (`http://polymaps.org`) is used to display run tracks on a satellite map and crossfilter (`http://square.github.io/crossfilter/`) supports exploring multi-variate data sets using coordinated views. Similar to FastBit on the back end, crossfilter efficiently identifies samples satisfying a boolean range query and supports interactive updates to multiple coordinated views. HTML5 provides the *canvas element* used for scriptable rendering of bitmap images, e.g., for false color plots.

3.1 Workspace Manager

Use Case: The first step in analyzing data is to identify runs of interest, e.g., find runs that were collected using a specific platform, provide data for a particular sensor system or were collected in a time period of interest. It is also convenient for analysts to maintain a list of "active" runs that they are currently working on. The *workspace manager* (Figure 3) provides filters (left hand side) to sub-select runs of interest based on platform, sensor system or a date range. As the analyst fixes values for these filters, the run list in the center is continuously updated. The analyst can then add runs from this list to the active set (right hand side)—i.e., a list of runs currently of interest that persists between sessions.

Implementation: The workspace manager is implemented mostly in JavaScript running on the analysts browser, using Bootstrap for the UI and jQuery for selection and filtering.

The workspace manager only requires access to a subset of information about a run, i.e., a list of the names of all runs along with collection time window, platform, and sensor system, which allows analysts to quicly identify runs of interest. The workspace manager obtains a list of this information from the SQL database on the back end via a REST query and creates the interface—including selectable filter values—and updates the list of runs matching the current filter criteria dynamically. More detailed information, such as a full list of data sets available and their time, is generally not necessary for run pre-slection and can be omitted at this step. As a consequence, the concept of a workspace also allows us to limit transferring detailed information about individual runs for the *data browser* and *waterfall plot*. Only the runs in the active set i.e., a subset of perhaps up to twenty runs out of the potentially thousands of runs stored in the database is accessed—reducing data transfer times and improving response time significantly.

3.2 Data Browser

Use Cases: After populating the workspace with one or more runs of interest, the *data browser* supports examining these runs in more detail, such as displaying the exact path along which data was collected overlaid on a satellite map view for spatial context. Furthermore, it is possible to select additional data sets and color the path based on these data sets, such as altitude in Figure 4. This type of display supports a quick evaluation of data collection coverage and a quick "validity check" of additional data sets to identify sensor drop outs, for example.

The main purpose of the data browser is to aid the analyst identifying data to either *query* data subsets relevant for further analysis tasks —see Section 2.2— or to identify runs that are of interest to be analyzed

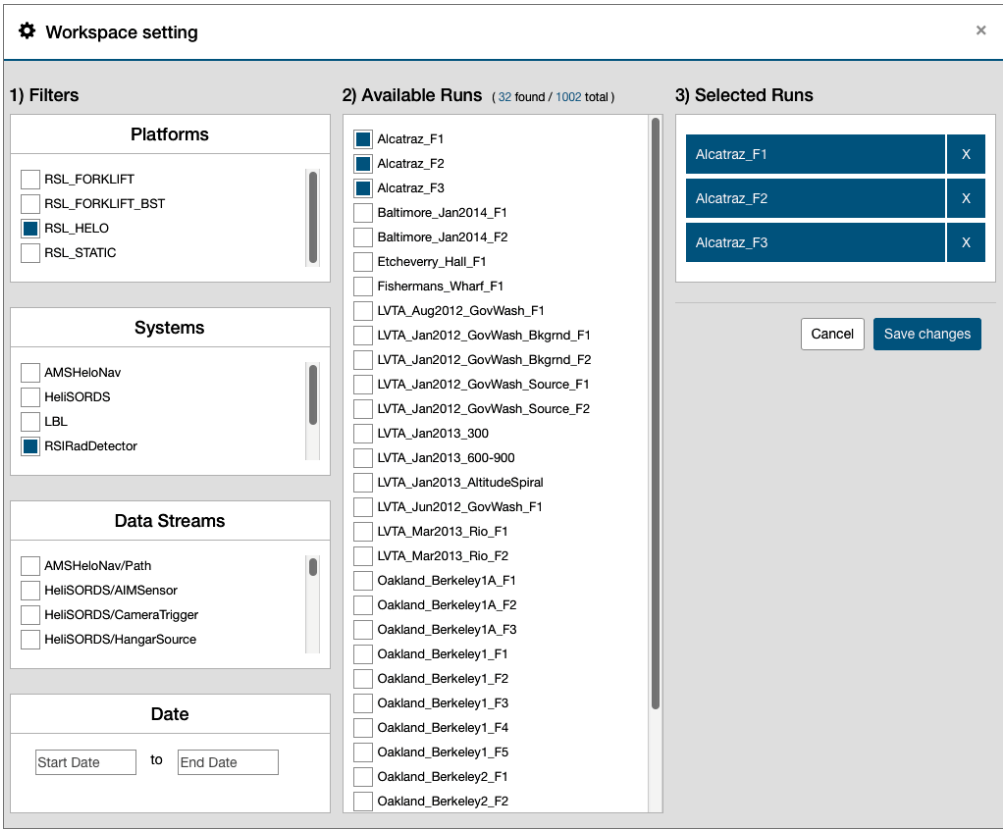


Figure 3: The *workspace manager* supports searching for runs of interest. Controls on the left specify filters for runs, a list in the center shows all runs satisfying the filter criteria, and a list on the right displays the runs currently in the analyst's workspace. Filtering, e.g., runs collected by helicopter ("RSL_HELO") using an older version of the gamma-ray detector ("RSIRadDetector") shows the corresponding list of runs in the center. By selecting these runs via the checkbox to their left, they are added to the analyst's workspace.

via programmatic connection. This is accomplished by providing immediate visual feedback of the content of the runs. For this purpose, the browser combines multiple coordinated views—a single satellite map view shows the path taken during the run and multiple histogram views show the distribution of data set values—using the concepts "brushing and linking [2, 9]," i.e., highlighting the data portions selected in one view in all other views.

To display information about an arbitrary number of data sets, the data browser displays histograms of their data distribution. An analyst can add new histograms for any data set containing scalar data. For example, Figure 4 shows histograms for altitude and the velocity component into the "northing direction." To formulate a query, the analyst can choose a rectangular region—corresponding to latitude and longitude ranges—in the map view and/or pick value ranges in the histograms.

Since all views are linked, selecting a value range in one histogram highlights the corresponding samples in the current histogram view as well as in the map view by graying out all non selected samples. The run path is shown in gray for portions outside the current query and red or colored according to a variable for the selected portions. All other histograms show the distribution only for data set values in the current selection.

In the center panel of Figure 4, the analyst has picked a value range in the "altitude" data set of approximately 700 to 1000 meters. In the map view, non-selected portions of the path are rendered gray, while parts falling inside the query are shown in color. In all histograms, selected data portions are shown in color—red or colored by value—and unselected regions are shown in gray. This coloring is updated while the analyst performs the selection providing immediate feedback as to whether the query selects any data at

all and what the distribution of currently selected data values are. For example, selecting an altitude range changes the distribution of the "velocity in northing direction," and that histogram is updated continuously throughout any selection changes.

A query string is constantly updated to correspond with the user's interactive actions. After finalizing the query, the analyst can select data sets for download (right panel of Figure 4) and submit the query to the back end. The back end will then produce an HDF5 file containing all selected data sets with data values falling inside the query.

Implementation: The JavaScript front end of the data browser uses (i) Polymaps to show the run path along which data was collected on a map, (ii) D3.js to display histograms and (iii) crossfilter to link these views together. Initially, the data browser submits a REST query to the back end requesting (i) a list of all data sets along with their their properties—i.e., their name, dimension, data type and size—and (ii) the run path—a list of longitude, latitude, altitude and time stamp values. The data set list is used to create menus for adding new histograms, coloring the run path by a data product and selecting data products for download in a query. Dimension and data type are used to down-select which data sets can be used to create histograms or as path color.

To reduce the amount of primitives drawn in the map view and to increase rendering speed, the data browser accumulates multiple latitude, longitude, time samples into line segments representing a portion of the path. Our current implementation uses a simple approach that accumulates samples along the path until a segment exceeds a specified length in the longitude/latitude dimensions or a specified maximum duration. The value of other data sets for the segment—initially only altitude—is computed as the mean of all sample values belonging to the segment.

The crossfilter library enables interactive updates of the multiple coordinated views by providing a means to quickly identify segments satisfying range queries. The data browser create a crossfilter data structure and inserts all segments into it, each data set constituting a "dimension" of the multi-variate data set. Selections within the map view or any histogram are translated into value ranges for each dimension data set. Crossfilter makes it possible to quickly identify the segments that satisfy this range query as well as compute histograms of the value distribution of each dimension. Histograms are displayed using D3.js, which also handles detecting user interactions like selecting a range in the histogram.

If the analyst requests displaying—either as run path color or histogram—a data set not available in the browser, the front end requests it from the back end using a REST query. The back end returns data values along with their time stamps. Using the time stamps to sync the data set with the flight path, data values are added to the appropriate flight segments. Once all data values are added to the segments, the average value for each segment is computed and the cross filter data structure updated.

3.3 Waterfall Plot

Use Case: Gamma-ray sensors detect gamma rays interacting within a detector as discrete events, recording the energy deposition and the time of the event. A waterfall plot is a two-dimensional histogram of gamma-ray events binned in both time and energy that allows the user to search for and examine transient spectral features during a run. The spectrum can be displayed as a false color plot of the two dimensional histogram with energy and time as axes (Figure 5). Bin sizes along these axes control the resolution and amount of smoothing for the histogram, making it possible to adapt it to the analysis task.

The gross count rate is plotted along the vertical axis and the gamma-ray spectrum is plotted along the horizontal axis. To display the large range of count rates, the false color plot shows the logarithm of count events. If there are no major spectral changes during a run, then the waterfall plot will appear to be made of vertical bands, where each vertical band is at the energy of a major background gamma-ray line, see Figure 6(i). If there are modulations in the overall count rate but no major spectral changes, then the waterfall will also appear to have horizontal bands, see Figure 6(ii). For very strong spectral anomalies, horizontal bands will appear only over a portion of the spectrum but not the entire energy range.

Similar to the data browser, rendering the waterfall plot requires us to address some challenges. Precomputing the spectrum is not a feasible approach since analysts might need to change the bin size in both time and energy dimension to study specific data of interest or subset of detectors that is of interest might

change. All data sets in the BDC are indexed, and FastBit supports fast computation of spectra from indexed files. Using FastBit/FastQuery, this computation takes on the order of seconds to a minute.

Consider one example of how a user would interact with the waterfall plot, see Figure 5. The gross count rate along the right axis might show a brief but large increase (Figure 7), and the user may want to inspect the data around the increase to see if further investigation is warranted. Using the waterfall, the user can zoom in to the region around the feature (either through keyboard input or dragging a box with the mouse), see Figure 7(i) and see greater details in the spectrum and count rate. For example, the user might notice an increase in counts below about 700 keV, which could mean there was a Cs-137 source present, see Figure 7(ii). They may then decide to download that particular dataset for more detailed analysis or to conduct an analysis remotely within the BDC server infrastructure.

Another use of the waterfall plot is to search for anomalies in particular spectral regions (see Figure 6). The user can input a range of energies—say, 620 to 700 keV to search for Cs-137 features—and then update the waterfall to see if there are any increases in the gross counts in that window. Those regions can then be further investigated.

Implementation: The front end displays the histogram as a false color plot using an HTML 5 canvas, see Figure 5. The histogram data from the back end is a 2D array with an event count per bin. The front end computes the logarithm of these events and displays a false color plot in an HTML5 canvas. It further sums up bins along time and energy axes to obtain plots of spectrum (bottom plot in Figure 5) and count rate over time (right hand side plot in Figure 5) respectively. Using d3.js and HTML 5 canvas, the front end supports zooming into the pre-computed histogram and panning the zoomed version. The spectrum and count rate plots are updated interactively. Thus, by moving only part of the computation to the back end, limited interaction is possible with the pre-computed spectrum. Furthermore, analysts can specify arbitrary bin sizes and obtain a spectrum in a reasonable time (a few seconds computation followed by a few seconds of data transfer).

To support interactive exploration of spectra, the system operates as follows. After the analyst has selected a run in the front end, it uses a REST query to identify for what data sets of a run spectra can be computed and what value ranges occur. The analyst then chooses minimum maximum ranges for energy and time and appropriate bin sizes for the histogram. The front end then requests a histogram from the back end using these parameters. The back end uses FastBit/FastQuery to compute the histogram in tens of seconds and sends it to the front end.

Many modern sensing devices do not consist of a single detector, but instead consist of an array of detectors, where each detects events individually. These detectors have different sensitivity depending on the gamma-ray direction, partially because they cover each other. Thus, by considering the detector id it is possible to infer the location of a source. To account for this, the data managed by the BDC contains a separate data set that specifies the sub-detector that recorded the event. To make use of this information, the waterfall plot also allows one to filter events by detector id before displaying the plot. It is also of interest to compute the spectrum for only a subset of detectors. The front end adds the ability to select any combination of detectors of interest. Again, through the use of FastBit/FastQuery it is possible to compute an appropriate histogram in little time.

3.4 Source Inject Interface

The BDC helps algorithm development to identify artificial radiation sources in natural background radiation. Several data set transformations serve this purpose. One of them is *source inject*, which simulates the spectrum of an artificial source on the background radiation. "Injecting" a source depends on various parameters such as location of the source, time at which the source is present, the isotope, and a model or measured data for the simulated source. The web front end aids the analyst in choosing these parameters. For this purpose, it provides a map view, similar to the data browser, wherein spatio-temporal information describing a run is displayed. In this map view, the analyst can select the location of the source and move it (see Figure 8). Furthermore, the analyst can specify the time window during which to simulate the presence of the source. To choose an appropriate time, the source inject UI colors the portion of the run path during which the source is active red and grays out the remainder of the path. In addition to this, it lists all other

parameters and allows the analyst to specify them. Once all parameters are set, the analyst can generate a file with the source description that then serves as input for the actual source inject process.

4 Related Work

Visualization for BDC combines web-based visualization, geographic information systems (GIS) and efficient data storage with the goal to present collected data in a geographical context.

Web-based scientific visualization has a long history including in bioinformatics. Driven by large initiatives to sequence the genome, this community focused early on data sharing—making sequences and annotations available to everyone to enable additional analyses. These efforts resulted in several web-based genome browsers, including the well-known UCSC Genome Browser [8]. More recently, the need for sharing data and fusing data sets from multiple sources resulted in integrated web-based systems such as KBase (`http://kbase.us`), that combine access management, the ability to run elaborate analyses as well as visualization and presentation.

Newspapers, such as the the New York Times, make wide use of web-based visualization. Interactive view graphs disseminate information and allow readers to explore them, e.g., to filter election results based on party membership, region, etc. and display the results immediately. This trend led to the development of JavaScript libraries—such as Data Driven Documents (D3) [3]—that simplify the mapping data to visual representation. These libraries also provide abstraction to implement user interactions. *Brushing and linking* is one common interaction pattern, in which selecting data items in one view highlights them in other views as well. We use many of the similar technologies to explore the nuclear background radiation data.

Interactive linking of views requires keeping track of active data items based on their values within multiple dimensions, a capability provided by the crossfilter library (`http://square.github.io/crossfilter/`). One example—given on their web-page—allows users to explore airline on-time performance. In this example, data dimensions consist of date, time of day, flight distance and arrival delay. Crossfilter is used to link multiple histograms views, each histogram showing the distribution of values in a single dimension. The user can filter flights by choosing value ranges in any histogram, e.g., selecting flights with a delay of more than an hour. Crossfilter enables updating all histograms immediately to show the value distribution of the current selection, allowing the user to identify relationships. We augment histograms with geographical map views and data filtering on the back end to support large data set size.

Geographic information systems (GIS) also make extensive use of web-based techniques. Review web pages like "Yelp!" and "TripAdvisor" show locations of points of interest on a map and augment position information with additional information, such as reviews and opening times. Other web pages support tracking seagoing vessels based on Automatic Identification System (AIS) information or airplanes using the Automatic Dependent Surveillance - Broadcast (ADSB) system. Finally, there are web-based information systems for weather, climate (`http://cal-adapt.org`) or other environmental information—like carbon, water, and energy fluxes (`http://ameriflux.lbl.gov`) on maps.

A key difference to these applications, is the size of tracks collected by the BDC and number of attributes associated with tracks. Due to these attributes, it is necessary to perform extensive data filtering on the back end server, to avoid overwhelming the web browser. What differentiates our system further from others is its use in forming queries and the use of scientific indices like FastBit [12] and FastQuery [4] on the back end to accelerate, e.g., histogram computation on-the-fly on the back end. Finally, the BDC stores a diverse range of data sets that need to be analyzed.

In defense applications, visualization can increase situational awareness; prior work focuses on showing positions without associated attributes in the context of evaluation data [5]. Other work focusing on web-based technologies to defense applications focused more on 3D technologies and authoring [11]. The development of radiation detection systems commonly requires plotting spectra similar to the waterfall plot in our system [7, 10], often in real-time.

5 Lessons Learned

As with large systems like the BDC cloud, there were many important lessons learned along the way in implementation of the BDC data cloud in general and the visualization system.

First, the initial design of the BDC database, file formats and stored meta data focused primarily on data ingest and data storage. Data access and use patterns were unclear during that stage in the project. Thus, the data ingest omitted information useful for visualization purposes. For example, while the database contains information about physical units (meters, keV, seconds), it does not contain information about time references, e.g., whether timestamps are stored in epoch time, i.e., as number of seconds that have elapsed since midnight UTC on January 1st 1970. This is an example of the type of information that is necessary for visualization in terms that are meaningful to a user. While it is possible to add this information separately to the database, such an approach leads to duplication and the risk of data inconsistencies. Similarly, based on units alone, it is often difficult to asses the type of visualizations that makes sense for a given data set. Again, a database design focused on how the data is used during analysis, in addition to how data is collected, would have simplified matters considerably and lead to a better overall data format design. However, data access patterns can be hard to assess when scientific data collections are intentionally designed to be exploratory in nature. Such collections can be conducted prior to creation of well-defined visualization and analysis methods.

Revising data storage based on evolving needs would be useful and make it possible to improve initial design choices. However, such a re-design is difficult for a project with active users and uptime requirements. Database storage layers such as NoSQL databases, i.e., next generation databases that model data with other means than the tabluar relations used in relational databases, offer opportunities in this direction. However, implementing such changes require resources and wider coordination across development, production and user teams, that makes it very challenging. Yet transition to such a database would consume substantial project resources. Thus, the stakes for "getting things right" in the first place are high. Continuous conversations about data access patterns over the life of the project and lifecycle of development might help mitigate these issues.

On the other hand, some initial design choices turned out to be highly beneficial during the later design and led to new approaches to visualization problems. For example, the FastBit indices were originally chosen as a means to speed up data filtering and access. However, FastBit's advanced histogram computation capabilities made it possible to compute histograms on the fly on the server backend, avoiding the need for storing pre-computed histograms and making it possible for users to adjust histogram bin sizes fairly interactively.

6 Conclusions and Future Work

In this paper, we describe the visualization system in the Berkeley Data Cloud that serves nuclear background radiation data. The visualization tools provide the gateway for data exploration and analyses in BDC. The tools help users identify and validate data sets of interest, identify regions and parameters of interest in the data sets, interactively explore the data and set up data for analyses. The visualization tools in BDC address important challenges related to real-time rendering and data management that is required for big data visualization. Our work is exemplar that demonstrates the use of web-based technologies for real-time visualization of large data sets of interest to the defense and homeland security community. In future work, we plan to develop advanced visualization of 3D and better interactive visualization and collaboration on the data through the web-based user interface.

As of this writing, we are in the process of obtaining permission by Lawrence Berkeley National Laboratory, the Department of Energy (DOE) and Department of Homeland Security (DHS) to release the BDC software publicly. Once we have this permission, the system will be available for download at `http://bdc.lbl.gov/#software`.

Acknowledgments

Some of this work was performed under the auspices of the US Department of Energy by Lawrence Berkeley National Laboratory under contract DE-AC02-05CH11231. That work was funded by the National Nuclear Security Administration, 382 Office of Defense Nuclear Nonproliferation research and development. Additional portions of this work has been supported by the US Department of Homeland Security, Domestic Nuclear Detection Office, Under IAA HSHQDC-11-X-00380. This support does not constitute and express or

implied endorsement on the part of the government. We thank the anonymous reviewers for their comments, which helped us to improve this paper.

References

[1] Mark S. Bandstra, Timothy J. Aucott, Erik Brubaker, Daniel H. Chivers, Reynold J. Cooper, Joseph C. Curtis, John R. Davis, Tenzing H. Joshi, John Kua, Ross Meyer, Victor Negut, Michael Quinlan, Brian J. Quiter, Shreyas Srinivasan, Avideh Zakhor, Richard Zhang, and Kai Vetter. Radmap: The radiological multi-sensor analysis platform. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 840:59–68, 2016.

[2] Richard A. Becker and William S. Cleveland. Brushing scatterplots. *Technometrics*, 29(2):127–142, May 1987.

[3] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. D3: Data-driven documents. *IEEE Transactions on Visualization and Computer Graphics (Special Issue: Proceedings of IEEE InfoVis)*, 17(12):2301–2309, December 2011.

[4] Jerry Chou, Kesheng Wu, and Prabhat. FastQuery: A general indexing and querying system for scientific data. In *Proc. International Conference on Scientific and Statistical Database Management (SSDBM)*, pages 573–574, Portland, OR, United States, 2011.

[5] Eliot Feibush, Nikhil Gagvani, and Daniel Williams. Visualization for situational awareness. *IEEE Comput. Graph. Appl.*, 20(5):38–45, September 2000.

[6] The HDF Group. Hierarchical Data Format, version 5, 1997–2016. http://www.hdfgroup.org/HDF5/. Accessed September 21, 2016.

[7] Christopher Joines, Michael Reed, Ron Guise, Justin Schmidhofer, Ray Yost, Matt Kiser, David Schwellenbach, Krikor Hovasapian, Paul Ainsworth, Ken Braithwaite, and James Essex. An extensible architecture for supporting next-generation aerial radiological measurements. In *Nevada National Security Site Site-Directed Research and Development Fiscal Year 2012 Annual Report*, pages 159–167, Las Vegas, NV, United States, 2012. DOE/NV/25946–1722.

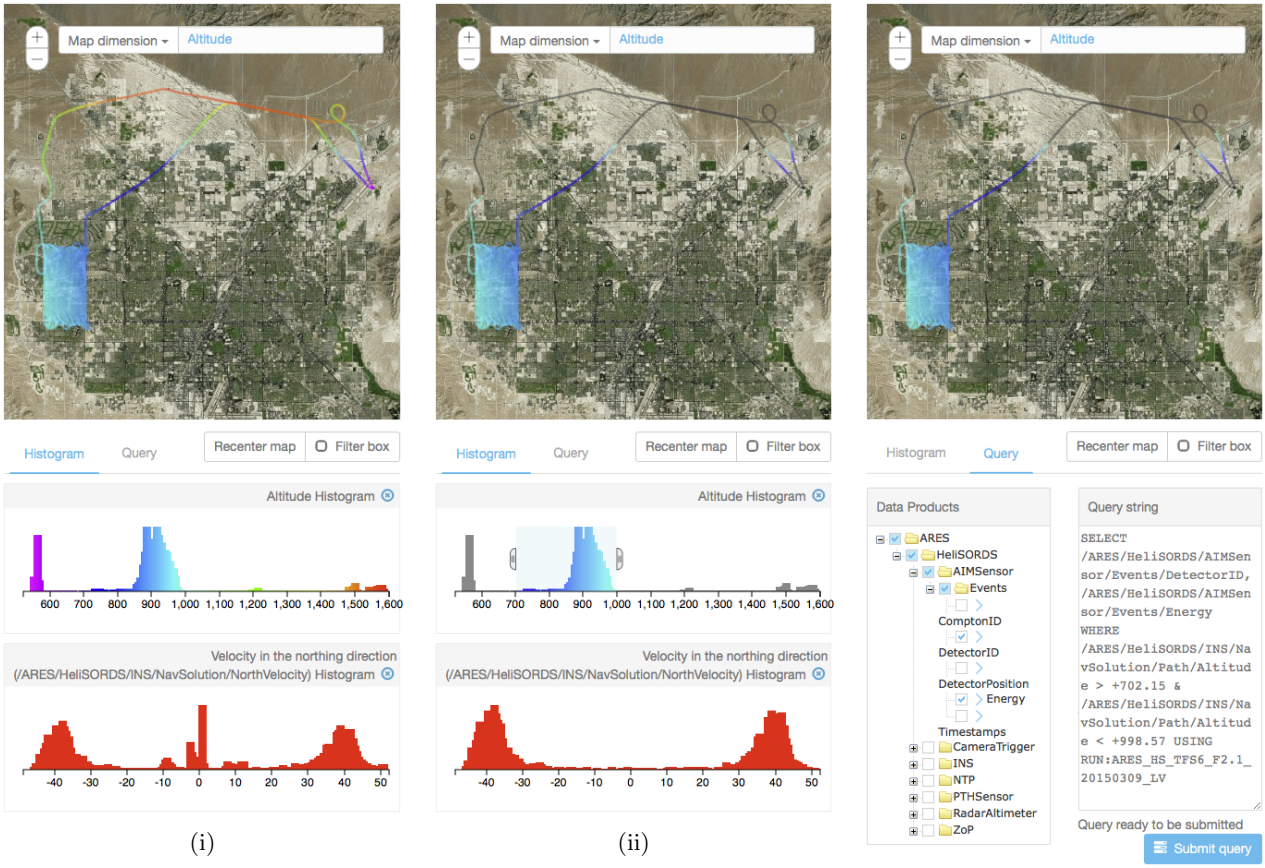
[8] W. James Kent, Charles W. Sugnet, Terrence S. Furey, Krishna M. Roskin, Tom H. Pringle, Alan M. Zahler, and David Haussler. The human genome browser at UCSC. *Genome Research*, 12(6):996–1006, June 2002.

[9] R. Kosara, G. N. Sahling, and H. Hauser. Linking scientific and information visualization with interactive 3D scatterplots. In *Short Communication Papers Proceedings of the 12th International Conference in Central Europe on Computer Graphics, Visualization, and Computer Vision (WSCG)*, pages 133–140, Plzeň, Czech Republic, 2004.

[10] Sanjoy Mukhopadhyay, Richard Maurer, and Paul Guss. Rapid response radiation sensors for homeland security applications. In *Radiation Detectors: Systems and Applications XV, Proc. SPIE*, volume 9215, pages 921504–1–921504–8, San Diego, CA, United States, 2014.

[11] Shane Nicklaus, Doug Horner, Curtis Blais, and Don Brutzman. Web-based 3d technology for scenario authoring and visualization: The SAVAGE project. In *Proceedings of Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC) 2001*, Orlando, FL, United States, 2001. `http://www.dtic.mil/get-tr-doc/pdf?Location=U2&doc=GetTRDoc.pdf&AD=ADA422086`.

[12] K. Wu, S. Ahern, E. W. Bethel, J. Chen, H. Childs, E. Cormier-Michel, C. Geddes, J. Gu, H. Hagen, B. Hamann, W. Koegler, J. Lauret, J. Meredith, P. Messmer, E. Otoo, V. Perevoztchikov, A. Poskanzer, Prabhat, O. Rübel, A. Shoshani, A. Sim, K. Stockinger, G. H. Weber, and W-M. Zhang. Fastbit: interactively searching massive data. *Journal of Physics: Conference Series*, 180(1):012053, 2009.



(i) (ii) (iii)

Figure 4: The *data browser* visually supports formulating queries. It can display the path and supports coloring by any scalar data set—such as altitude in the figure—to show data in a spatial context (i). To explore the relationship between data sets, the data browser can display histograms of data sets—such as altitude and velocity (northing direction in figure). The analyst can formulate queries by selecting a rectangular region in the map view or ranges in the histogram. All views are linked by means of highlighting. For example, if the user selects an altitude range between approximately 700 and 1000 meters, all portions of the flight path outside that range are "grayed out" and the histograms for all other data sets show the value distribution of the selection instead of the entire data set (ii). After selecting an appropriate data sub-set, it is possible to download the data from the BDC via an appropriate query (iii).

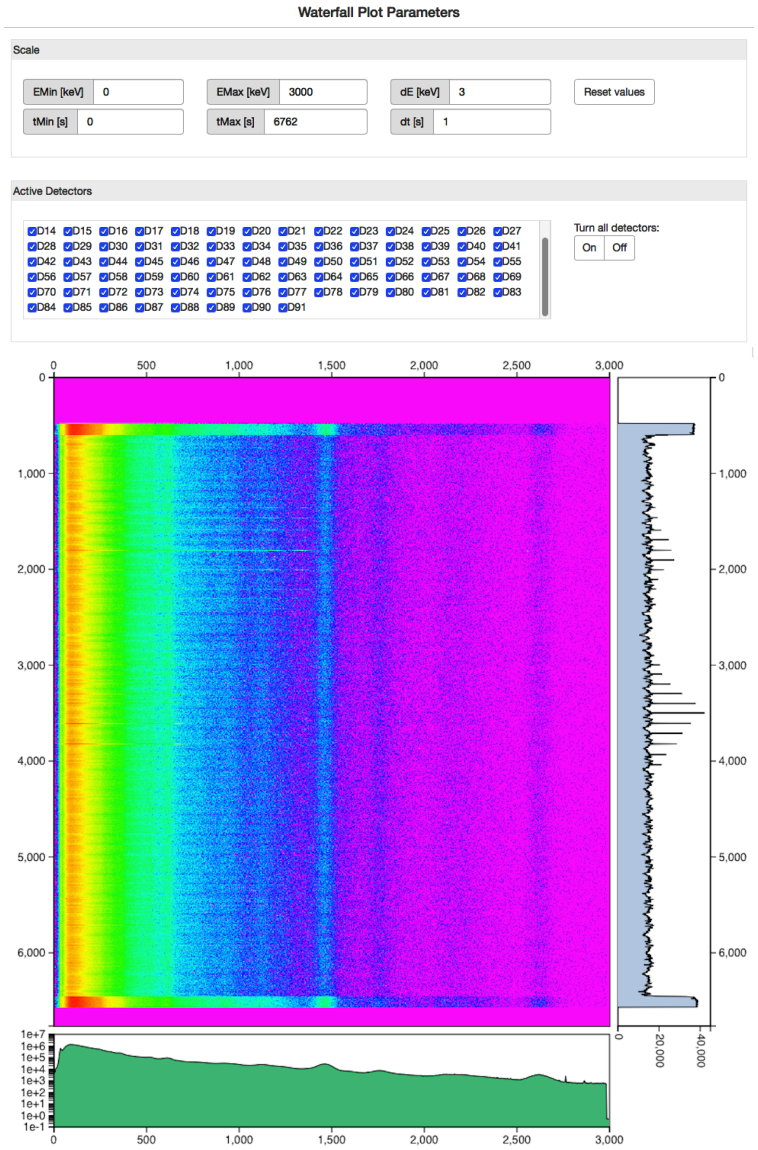
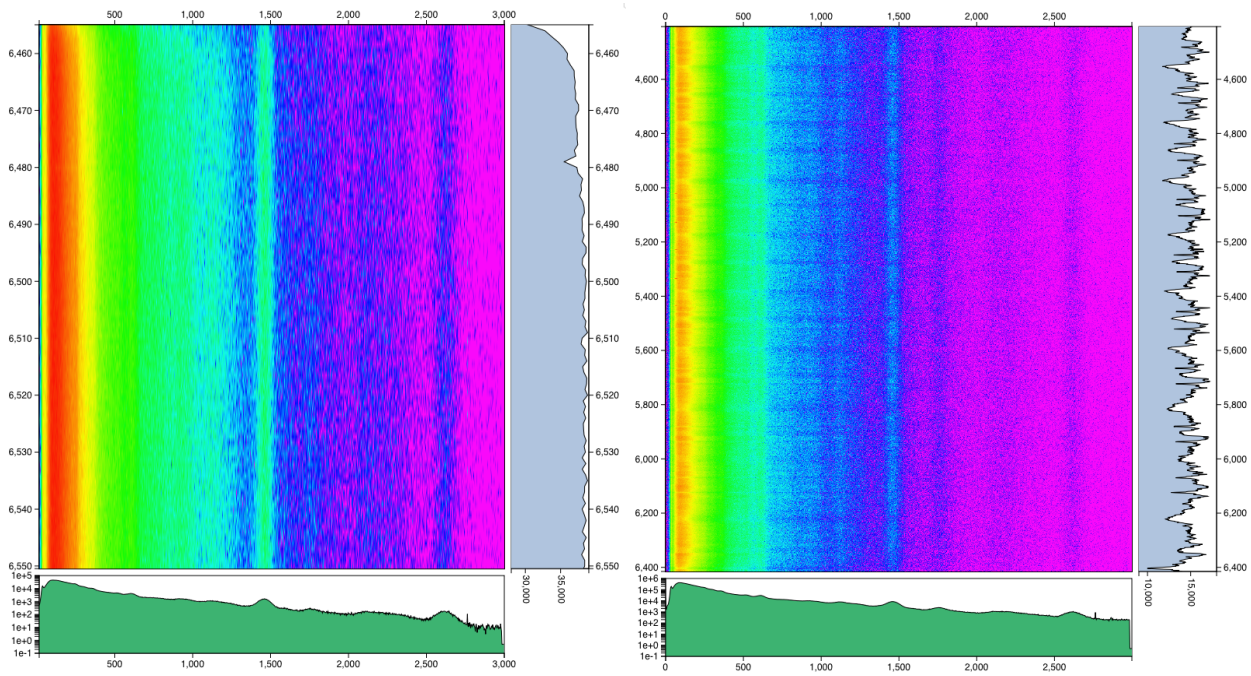
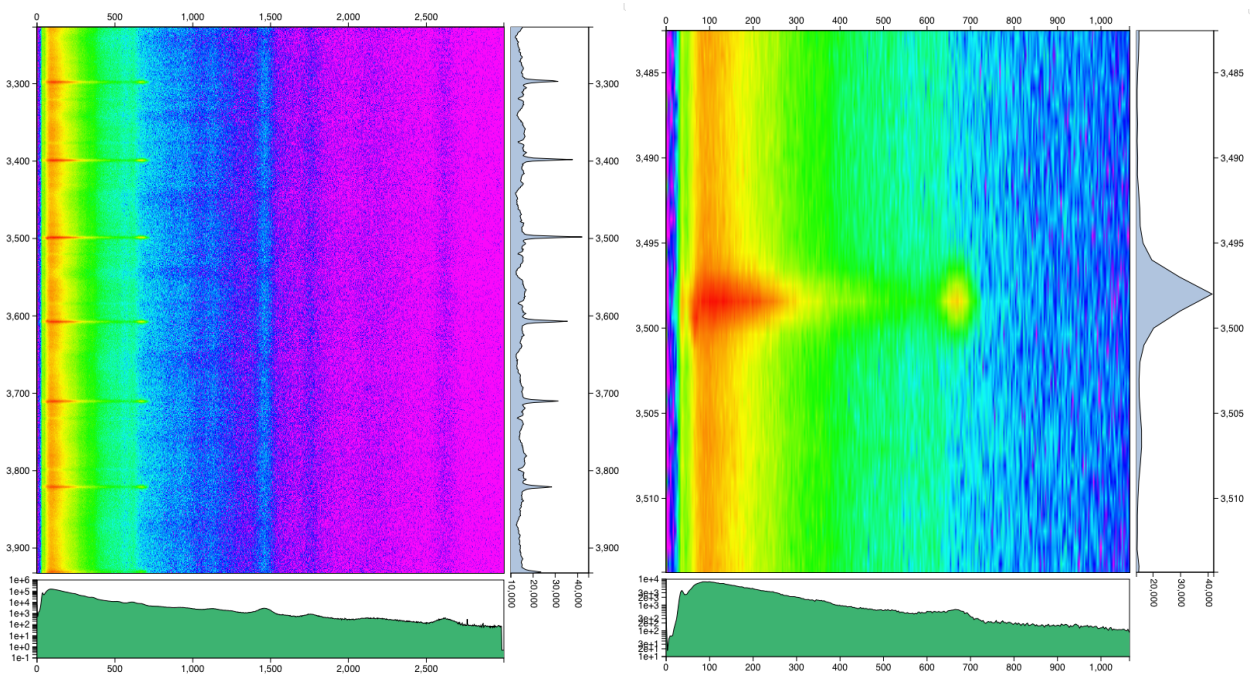


Figure 5: After specifying parameters, such as bin size and active detectors (top), the resulting waterfall plot shows the spectrum as false color plot (in the center), accumulated spectrum over all time steps (bottom) and the gross count rate over time (right hand side). This particular run featured numerous passes adjacent radiological sources in a controlled environment. If these sources were not under regulatory control, they would pose a threat to public health.



(i) (ii)

Figure 6: If there are no major spectral changes during a run, then the waterfall plot will appear to be made of vertical bands, where each vertical band is at the energy of a major background gamma-ray line (i). If there are modulations in the overall count rate, e.g., due to the helicopter turning around, but no major spectral changes, then the waterfall will also appear to have horizontal bands (ii).



(i) (ii)

Figure 7: Using the waterfall plot to examine the spectrum shown in Figure 5 in more detail. (i) Zooming in on some of the spikes in the gross count rate. (ii) Detailed spectrum of one of the spikes, strongly suggesting there was a Cs-137 source present.

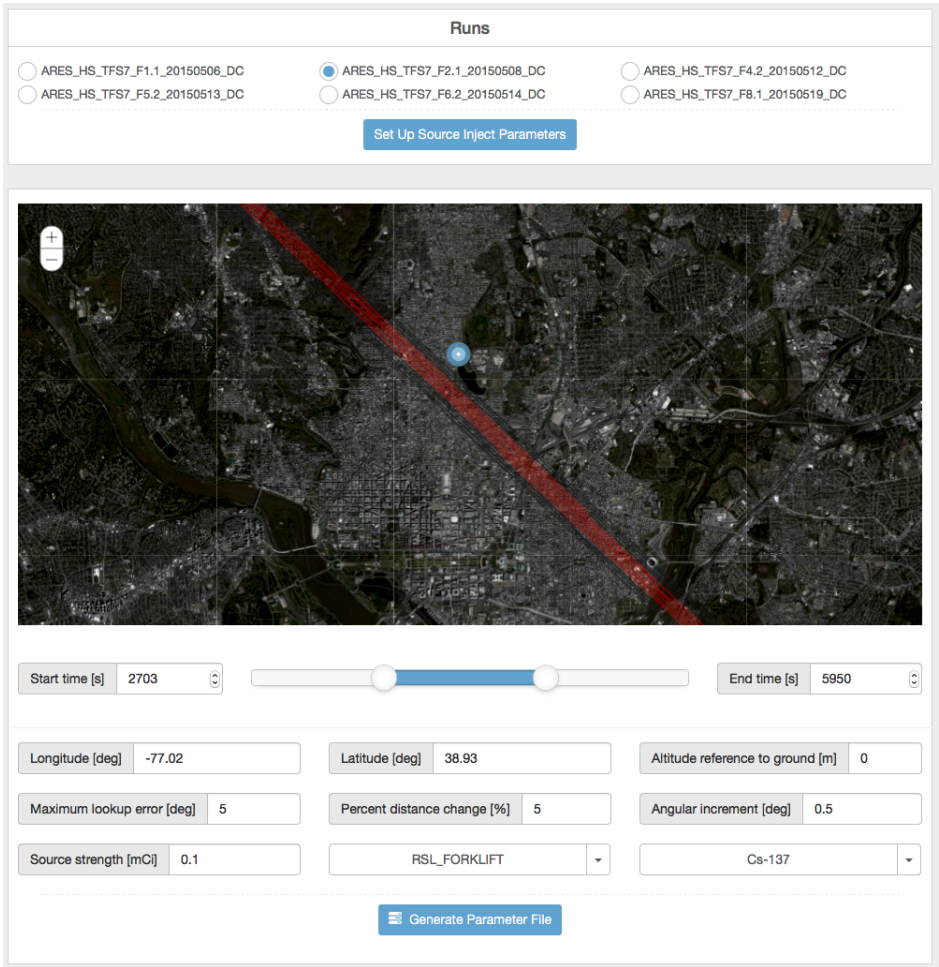


Figure 8: The source inject interface supports visual specification of parameters for injection of a simulated, artificial source into the data stream.