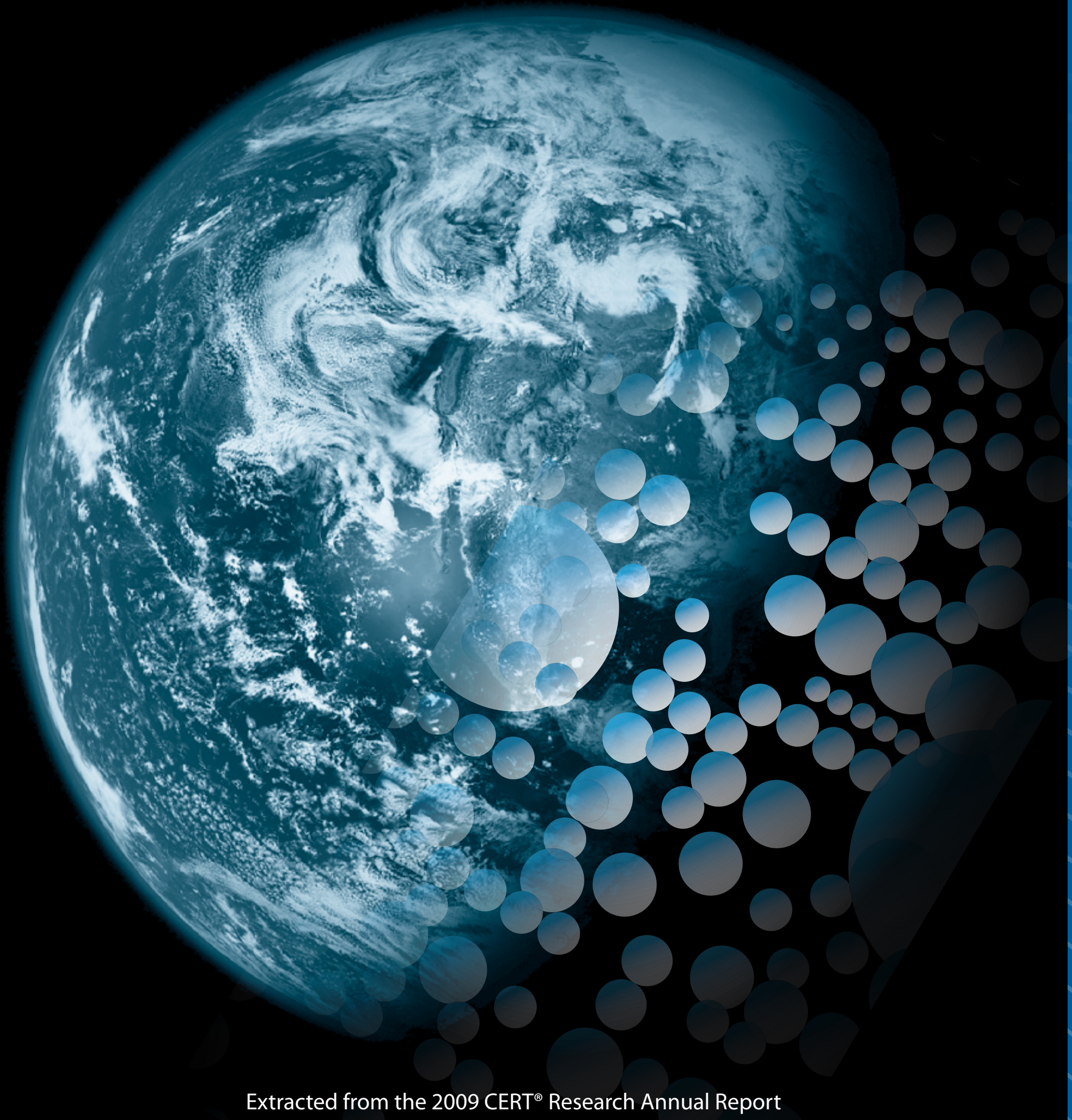
Rayon: A Unified Framework for Data Visualization

Phillip Groce



Rayon: A Unified Framework for Data Visualization

Introduction

Data visualization summarizes large volumes of data and represents this data pictorially. Data visualization is used in a wide variety of applications, but visualization techniques that are effective in one application can often be used as well or better in another application. When organizations depend on good data visualization, a unified visualization capability will often increase that effectiveness; this is especially important if an organization relies on internal experts to create new visualization techniques appropriate to their environment. The Rayon visualization toolkit was developed to augment large-scale network analytic information, and to improve the visualization capability and productivity of analytic operations by making it possible to share visualization techniques between applications.

Problem Addressed

To be effective, network security analysts must quickly analyze large volumes of network monitoring data. Network analysts often specify custom visualizations in order to meet this challenge. These visualizations are applied to data to produce graphics, which highlight aspects of network activity the analysts regard as important.

Graphics may be produced for several purposes, for example

- Exploration: An analyst generates a graphic to better understand a specific problem. The graphic is part of an iterative process of understanding the data. An interactive graphic may be useful to iteratively "walk through" the dataset.
- Explanation: An analyst generates a graphic to illustrate a point. The data for the graphic is well understood and carefully chosen. The graphic may be heavily annotated. The graphic may be printed, or included in a slide presentation.
- Validation: A scheduled process generates a graphic periodically as part of ongoing reporting and situational awareness. Analysts have expectations about the data, and the graphic either validates or highlights violations of these expectations. When expectations change, a series of validating graphics may document a transition from one set of expectations to another.

These use cases have diverse requirements concerning (among others) how quickly the graphic can be generated; whether the output can be saved or shared; and how much user attention is required to generate them. However, core visualization techniques such as scatterplots, bar charts and line plots can often be shared within these use cases. When an organization develops a new visualization technique or useful combination of existing techniques, it can be most effectively used when it is available for all these use cases.

Several software packages provide visualization capabilities. Examples are R [1], Matlab [2] and Microsoft Excel [3]. Sometimes visualization is provided as part of a larger analytic tool. Other tools exist only to do visualization. All of these packages simplify the process of data visualization. However, different packages have different strengths, and using multiple packages presents problems:

- No common language exists that applies to all visualization tools. Visualization techniques developed in one tool cannot be migrated to another without additional software effort. Similarly, different tools may export graphics in different ways. Some tools may support highly interactive graphics, but little to no export functionality.
- Visualization tools often depend on additional software. These dependencies differ from package to package, and may not be installed or available. The additional packages may complicate configuration management and lengthen time to deployment (e.g., by requiring additional change control requests).
- Default options are inconsistent between tools.

Ideally, an organization's visualization capabilities would be integrated to provide a consistent user interface and portable visualization techniques.

Research Approach

The Rayon visualization toolkit has been developed to provide a single platform that can be used to generate graphics in data exploration, explanation, and validation applications. Rayon provides an application programming interface (API) in the Python [4] programming language for specifying visualizations, and a facility for generating graphics from these specifications using different low-level software packages. Rayon can currently render static graphics using the Cairo [5] graphics library or interactive graphics using the WxWidgets [6] GUI toolkit.

In Rayon, a visualization developer (e.g., a network analyst who wishes to view a type of data in a particular way) specifies a visualization by creating an object called a *Plot*. Rayon comes with several common types of Plots. Plots can be overlaid onto each other or tiled; visualization developers may therefore compose more complicated Plots from numerous simpler Plots. For example, a line plot of trended data may be laid over a scatterplot of raw data.

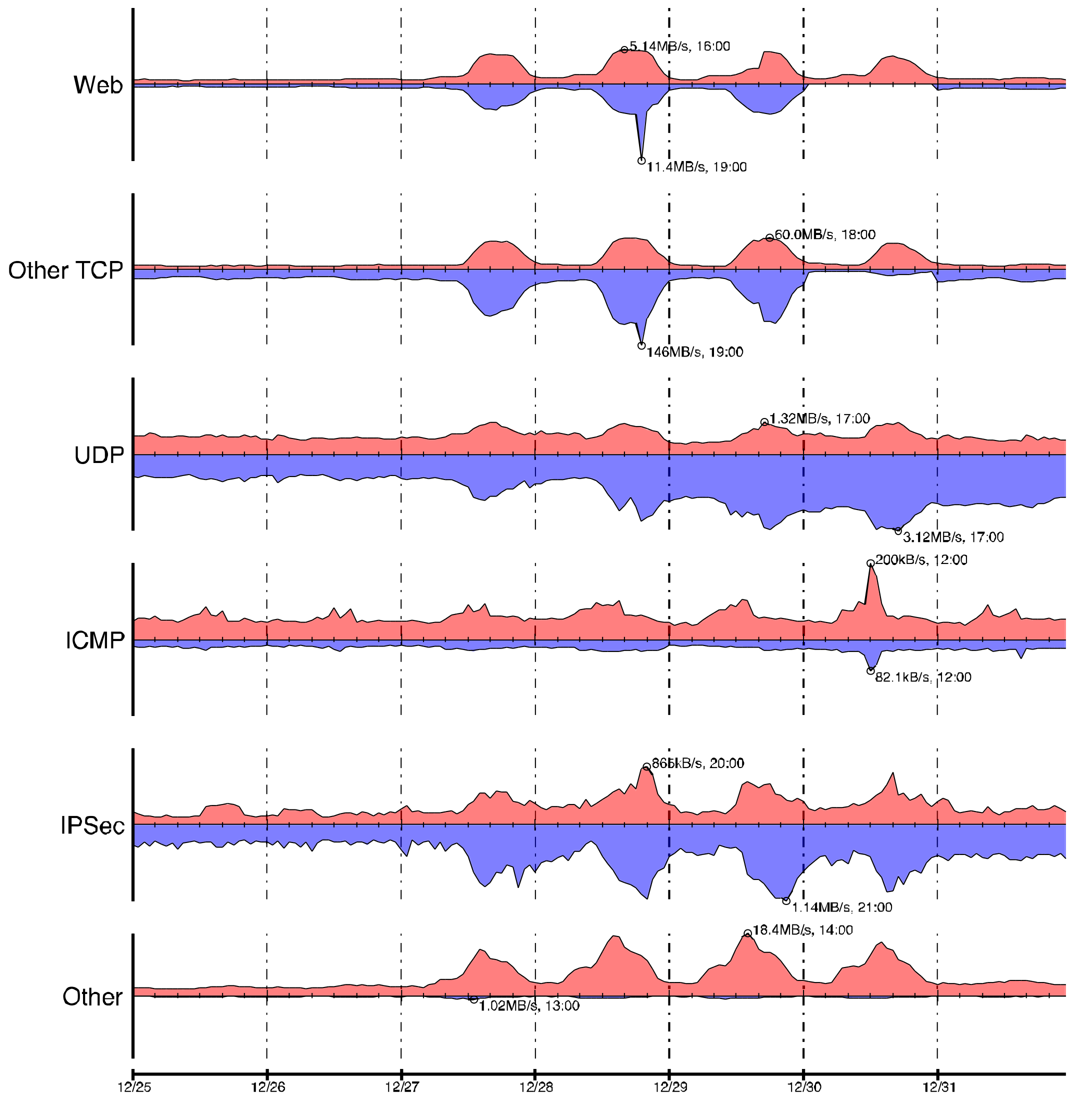


Figure 1: Multiple time series plots generated with Rayon, showing network traffic volume broken down by application and protocol. Incoming traffic is red, and outgoing traffic is blue. All the plots share the same time and volume scales, so volume can be compared across them.

Once a Plot has been created, it may be used to generate a graphic without having to understand how the Plot works. It can also be used within a larger Plot, again without having any additional knowledge. In this way, visualization developers can share and reuse their work. Figure 1 is a simple example of this. Different data are applied repeatedly to the same plot; the resulting graphics are tiled vertically to display an overall characterization of network traffic.

Engineers developing applications with visualization capabilities can write software that generates graphics from these visualization specifications knowing only the type of data it requires for input. If a visualization developer has defined events that the Plot handles, developers of interactive GUI applications can attach event listeners to them and provide interactive features. For instance, when an application end-user moves a mouse over a point in a scatterplot, the application can display a transient window containing additional information about the point.

Organizations benefit from Rayon by using or writing their own Rayon-enabled applications. Rayon comes with a set of UNIX command-line visualization tools. These tools may be used with arbitrary data, but they are designed to integrate well with the System for Internet Level Knowledge [7] (SiLK) network flow analysis tools. Each Rayon command-line tool generates a visualization based on data from a file (or UNIX standard input) and configuration information passed as command-line options. Integration is also planned with iSiLK [8], a graphical interface to the SiLK tools.

At minimum, Rayon requires a Python interpreter and one of the low-level libraries Rayon uses to generate graphics. When a dependency does not exist on a system, Rayon will, whenever possible, install with diminished capability rather than fail to install.

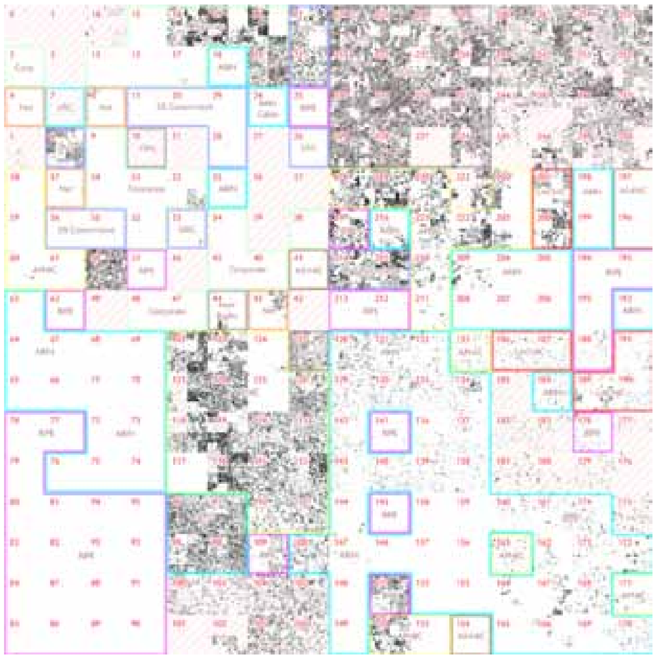


Figure 2: A visualization of network traffic generated using ryhilbert, a Rayon-enabled visualization tool that displays blocks of network addresses such that contiguous addresses are adjacent. Each point is a network block; blocks are shaded to indicate traffic volume, and colored borders are laid over the visualization to indicate IP allocations. Larger squares indicate larger networks; the red numbers in the "upper left" of blocks representing Class A networks show the first octet of those networks' addresses.

Expected Benefits

Rayon provides a visualization capability that can be used in applications across an organization. Visualization developers can create new visualizations which can generate graphics from any Rayon-enabled application, and support many different graphical use cases with a single codebase. Application developers can use the same visualization capability to support simple basic command-line tools, analysis scripts, and full-featured GUI and web applications.

2009 Accomplishments

Rayon has been incorporated into the most recent revision of an analytic tool that displays network traffic trends over time. One of the products of this tool is the visualization in Figure 1. The tool classifies network traffic by type and direction and presents the results as a set of parallel time-series graphics. The graphics share scales to make comparison easier.

Rayon has been used to implement two command-line tools. *ryscatterplot* generates scatterplots from the command line. *ryhilbert* generates graphics that represent IP networks in two-dimensional space, such that contiguous network addresses are adjacent to each other. An example is given in Figure 2.

2010 Plans

The focus of Rayon development for 2010 is to expand the number of Rayon-enabled applications. Rayon will be integrated into additional network analysis tools; one example is a tool characterizing network traffic from a set of IP addresses as using time series plots and sparklines. Command-line tools are planned to provide the user with access to time-series plots, bar plots, and line plots. Other command-line tools will be developed to perform simple analytics with obvious visual products, such as cumulative distribution functions.

In addition to expanded application support, the library will be expanded to provide additional plot types, such as matrix scatterplots and heat maps. To further support consistent styling across applications, a configuration interface is planned so that a user can provide custom styling information from a single source (e.g., a configuration file) to any Rayon-enabled application.

References

[1] The R Project for Statistical Computing. http://www.r-project.org/

[2] The MathWorks – MATLAB and Simulink for Technical Computing. http://www.mathworks.com

[3] Excel Home Page – Microsoft Office Online. http://office.microsoft.com/en-us/excel/default.aspx

[4] Python Programming Language -- Official Website. http://python.org/

[5] Cairo. http://www.cairographics.org/

[6] wxWidgets: Cross-Platform GUI Library. http://www.wxwidgets.org/

[7] SiLK. http://tools.netsa.cert.org/silk/

[8] iSiLK. http://tools.netsa.cert.org/isilk/