GRAPHENE: A JAVA LIBRARY FOR REAL-TIME SCIENTIFIC GRAPHS

K. Shroff, G. Carcassi, BNL, Upton, Long Island, New York, USA

BROOKHAVEN NATIONAL LABORATORY

U.S. DEPARTMENT OF ENERGY

ABSTRACT

While there are a number of open source charting library available in Java, none of them seem to be suitable for real time scientific data, such as the one coming from control systems. Common shortcomings include: inadequate performance, too entangled with other scientific packages, concrete data object (which require copy operations), designed for small datasets, required running UI to produce any graph. Graphene is our effort to produce graphs that are suitable for scientific publishing, can be created without UI (e.g. in a web server), work on data defined through interfaces that allow no copy processing in a real time pipeline and are produced with adequate performance. The graphs are then integrated using pvmanager within Control System Studio.

INTRODUCTION

The main aspect of graphene is that the painting is done on a buffer, instead of on screen directly. This means that all the painting operations can be done without a UI, for example in a web server, and on background threads.

Dataset definition are pure interfaces and are not coupled with any external type system. This allows graphene to read data from any source with no copy operations, issue that is critical while handling large datasets.

Since data changes dynamically, the auto-ranging has to take that into account to avoid the continuous stretching and shrinking of the axis. Graphene keeps both the current range of the dataset and the aggregated range, so that the axis can grow monotonically until they reach a stable size.

GRAPHS

The graphs currently available in graphene include the following:

Line graph

Scatter graph

Histogram

Bubble graph

Figure 1 shows some examples of the available graphs and interpolations algorithms.

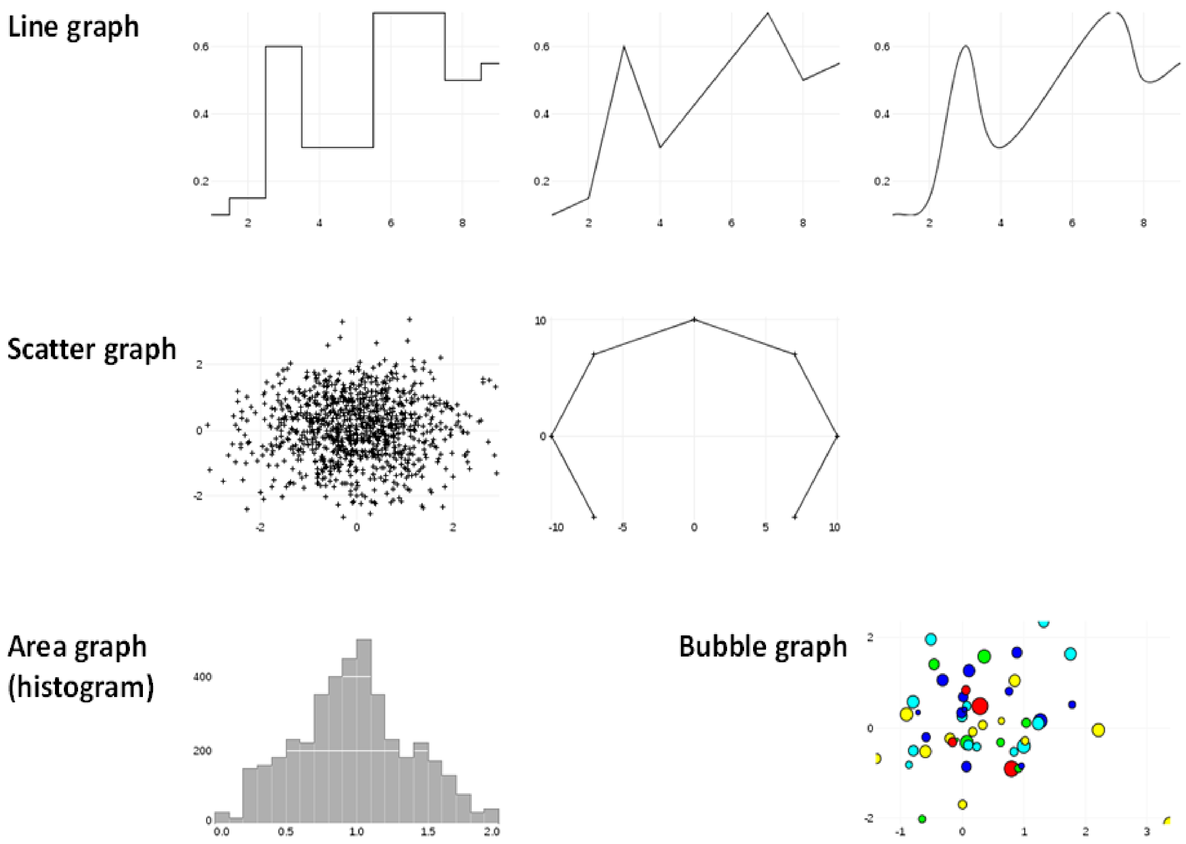


Figure 1: Example plots and interpolation algorithms available in Graphene.

PERFORMANCE

Given the ability of Graphene to plot on any thread, painting can be parallelized on modern multi-core systems. In the diagram, we show the number of graphs per second (600x400 histogram) on an Intel Core i7-840 Quad Core 1.87 GHz. One can see that the throughput increase linearly when increasing from one to four threads, and then saturates. With 8 threads, 3871 graphs per seconds are generated, which is more than enough to handle 3 or 4 graphs on screen with a fluid rate of refresh (200 graphs per second would be needed).

Additionally based on the size of the plot requested, Graphene automatically performs data reduction significantly improving performance. Fig 3. and Table 1. show the impact of graphenes data reduction on the performance of the Line graph, each pixel we draw the first, min, max and last values.

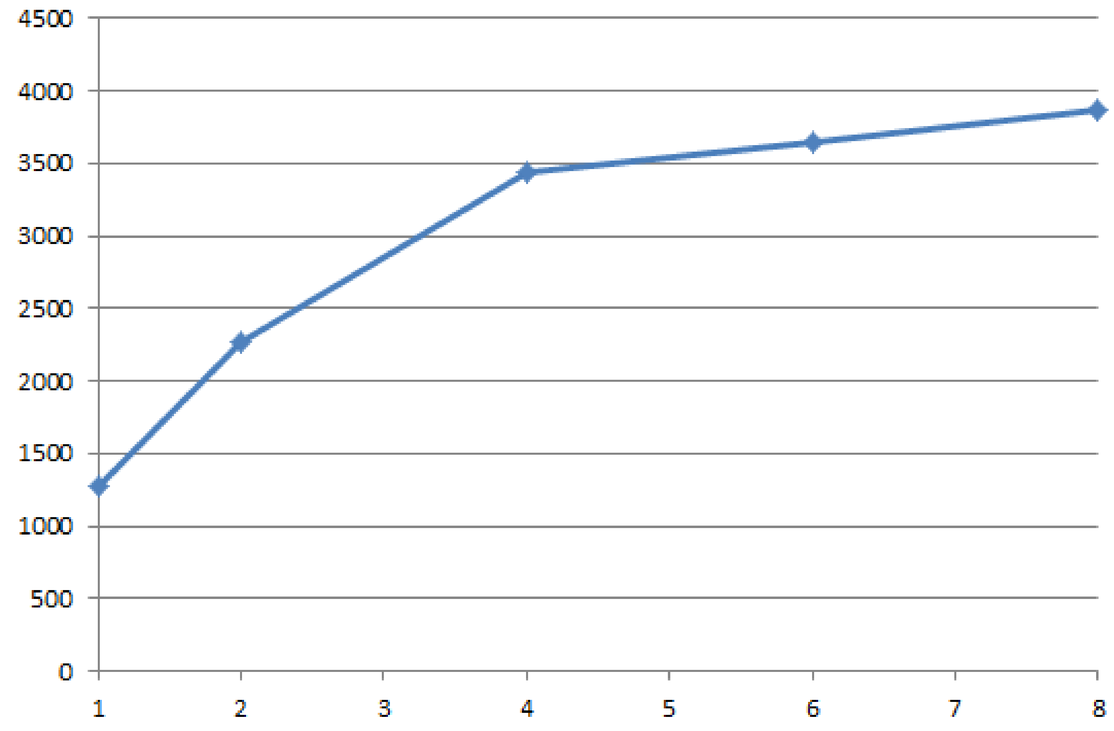


Figure 2: Performance of Graphene in a multi core environment. Number of plots per second vs number of threads.

LineGraph performance improved through data reduction, for each pixel draw 4 values: first, max, min, last

Displaying 100,000 points, 300x200

How many differences can you find?

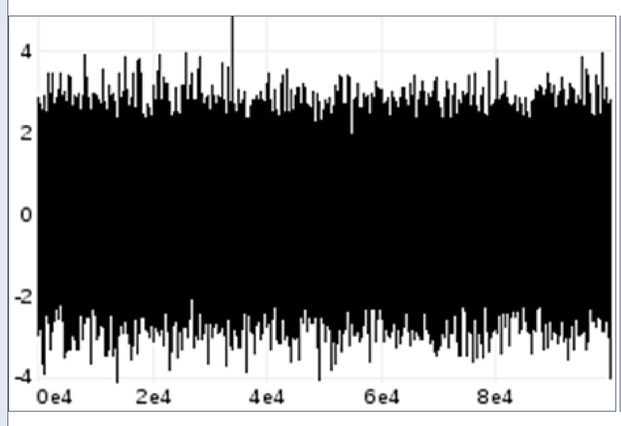


Figure 3a: Without data reduction - 255.4 ms

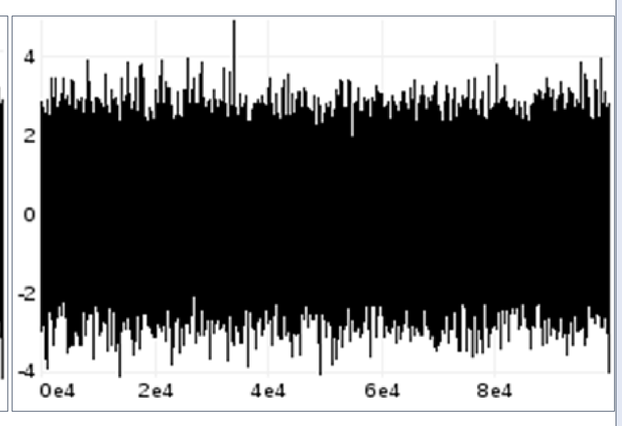
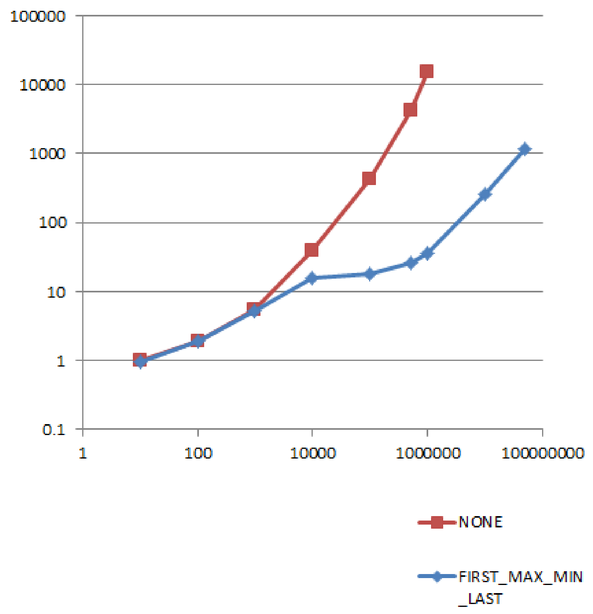


Figure 3b: With data reduction - 6.077 ms



N points	Time ms (no redux)	Time ms (redux)
10	0.986232349	0.980788919
100	1.924796332	1.872743481
1,000	5.553831552	5.345118646
10,000	38.57528235	15.73147583
100,000	431.8435542	18.28121567
500,000	4280.394044	26.20789778
1,000,000	15556.68846	36.36819781
10,000,000		257.451379
50,000,000		1178.120411

Figure 4a and 4b:Comparison of the performance for producing line plots with and without data reduction. Number of points plotted vs plotting time in milliseconds

CONCLUSION

While Graphene is still a work in progress, and we lack the appropriate resources for making its development proceed at a rapid pace, it is on the right track with the performance goals we set. Future work will also include exploration to provide JavaFX components out of the box.

REFERENCES

[1] http://pvmanager.sourceforge.net/

[2] G. Carcassi, Pvmanager & Graphene, EPICS spring meeting (2013)

[3] Control System Studio; http://controlsystemstudio.github.com

[4] http://graphene.sourceforge.net/