Performance in the Infragistics® xamDataChart™

An Infragistics Whitepaper

Published December 21st, 2010

1 Contents

1. Performance and User Experience

Everyone talks about user experience today – it is a key differentiator in the new economy. When most people talk about the user experience of their application, the conversation is generally focused on the aesthetic design of their application, which while important is merely one aspect of the overall user experience. Often people neglect to think about an area of user experience that can have just as strong of an impact on the overall experience of their application as the application's aesthetics **– application performance!**
At Infragistics, we take the user experience of your application very seriously, which means we not only focus on helping you provide a world class visual experience, but also the performance experience of our products in your application. In this whitepaper, you will learn how we approach the testing of our Silverlight/WPF xamDataChart and see the results we achieved in common scenarios, which support our claim of providing the fastest WPF/Silverlight chart on the market.

Why does performance matter so much in the overall user experience?

- Higher Customer Satisfaction
- Improved End User Productivity
- Resource Efficiency in Hardware and Bandwidth

The customer does not want to be, nor should they be bothered with the details of how an application is developed; they just care about their experience when they use an application. They want an application that simply "makes sense", which translates to an application that provides an intuitive user interface, and performs as the customer would expect – in other words – they shouldn't notice or think about how it performs. When you choose the Infragistics xamDataChart, you get a control that allows you to provide that "makes sense" experience to your end users, including the best performance on the market. And this is not at the cost of implementing some custom data provider – this is out of the box data binding to various data services and sources. No changes to your data access strategies.

2. Exceptional Performance Best Practices

During our product life cycle, we test our controls at various stages as new features are added or use cases updated. We follow a series of best practices in the **Planning**, **Configuration**, **Implementation** and **Review** of our performance tests to ensure the controls are tested in the correct scenarios and that the results we generate are accurate.

• Planning – During planning we determine what we are testing against (a previous build, a competitor, etc) and consider the options and tools available for performance testing on the specific platform. We also identify the specific scenarios that we want to test, and how we can create competitive tests that use as equal a configuration as possible.

• Configuration – In this phase we configure a testing environment that simulates as close as possible what customers are using based on the scenarios that describe the product requirements

• Implementation – We implement the tests outlined in the test plan and execute those many times to generate the performance statistics

• Review – During review, we evaluate the test results for consistency, looking for anomalies that can indicate a problem in the testing environment that could be affecting test results

Based on the above described best practices we implement and execute automated performance tests against each nightly build of our products. During the testing, performance results are automatically stored and reports are generated. This enables the development team to constantly monitor the performance through the development process.

3. Testing the xamDataChart

For NetAdvantage® for Silverlight Data Visualization and NetAdvantage® for WPF Data Visualization, the majority of our performance testing was focused on the xamDataChart because of its use in data-intensive applications and the large number of features it includes. Because we regularly talk to customers whose applications require chart controls that can load very large volumes of data at high refresh rates, when setting our performance goals and designing our tests we are able to consider those real-world use scenarios to help guide us. When testing the xamDataChart, we used a test environment as described below:

• A physical machine (non-virtual) - which would simulate a real life scenario.

• The test machine runs Microsoft Windows 7, CPU 2.13GHz, 4 GB of RAM.
• The test machine runs Internet Explorer® 8 as its default browser.
• Windows Update and anti-virus software are disabled.
• The test machine has Silverlight 4 (4.0.50917.0) installed.

To run the tests, we wrote code which ran all of the test scenarios 100 times. The scenarios ranged from simulated user clicks, drag-and-drop, to complete Remove, Add, and Edit data points operations. Each run used our latest nightly build and the most recent build of competitor controls available. To determine test results, we recorded the time and memory consumption of each operation, and then calculated the differences.
Memory consumption was calculated using the GC.GetTotalMemory function, and time was calculated using normal TimeSpan functions.

4. Performance Results

We developed test applications using the latest public releases of our and several different competitors' charts. These test applications are designed to ensure as much as possible that we compare apples to apples, and they simulate real-world scenarios found in the financial, insurance, and telecommunications sectors. We tried to achieve the maximum level of performance of competitors' products using publicly available guidance on their websites.

In this white paper we quote numbers Infragistics Silverlight xamDataChart™ and its best competitor chart from performance perspective and name it "Competitor I".
Both charts have common APIs between their Silverlight and WPF versions, with Silverlight charts delivering similar performance to their corresponding WPF versions.

The primary scenarios for charts identified the following areas as critical to the overall user experience of using a chart for data display and editing:

- Data binding
- Zooming/Scrolling
- Remove data points
- Add data points
- Edit data points
- Live data scenario

In this whitepaper, unless otherwise specified, we are showing the results of tests executed using data sets generated using a sinusoidal function.

We tested several types of series and found that in only a few of them (Area, Line, and Scatter) we could compare equivalent features versus our competition. This was due to the fact that in most cases our competitors didn't have these features.

4.1 Data Binding

To test the performance of basic data binding we created tests that measure the time needed to bind the xamDataChart using different series types and render up to the LayoutUpdated event. We believe this most accurately represents what an end-user experiences when waiting for data to load.

Table 1: Area Series chart binding

Type	Points #	Metric	IG xamDataChart™	Competitor I	
Area Series	1,000	Time duration in ms	211	1,725	717 % Slower
		Memory Usage Before in KB	1,801	1,447	19 % Less
		Memory Usage After in KB	1,977	5,981	202 % More
	1,000,000	Time duration in ms	617	1,733	180 % Slower
		Memory Usage Before in KB	31,036	31,517	1 % More
		Memory Usage After in KB	41,103	50,128	21 % More

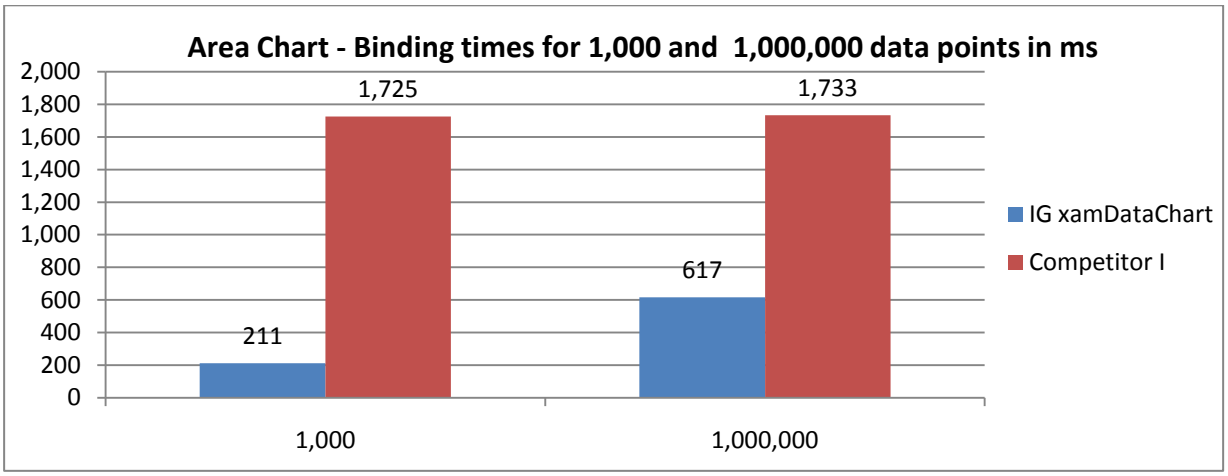


Table 2: Line Series chart binding

Type	Points #	Metric	IG xamDataChart™	Competitor I	
Line Series	1,000	Time duration in ms	152	1,332	776% Slower
		Memory Usage Before in KB	1,777	1,499	15% Less
		Memory Usage After in KB	2,073	7,371	255% More
	1,000,000	Time duration in ms	635	1,580	148% Slower
		Memory Usage Before in KB	31,031	31,523	1% More
		Memory Usage After in KB	41,064	52,127	26% More

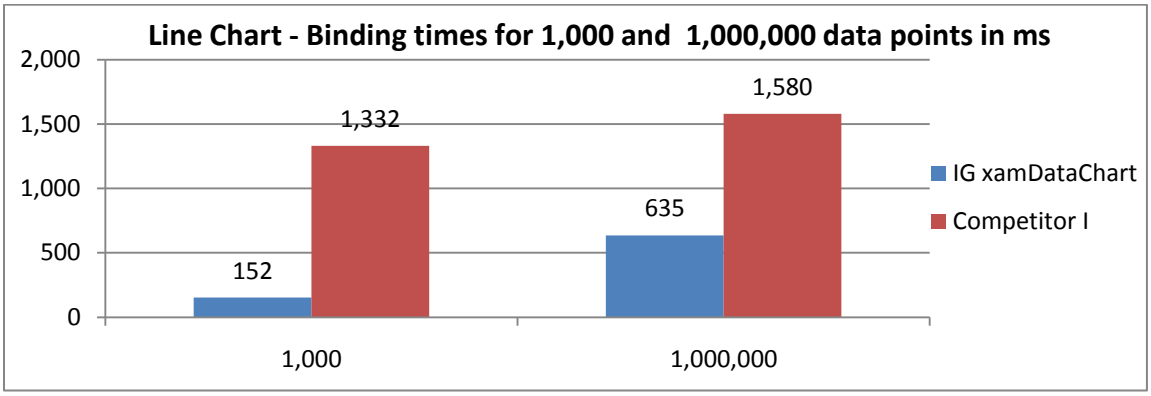
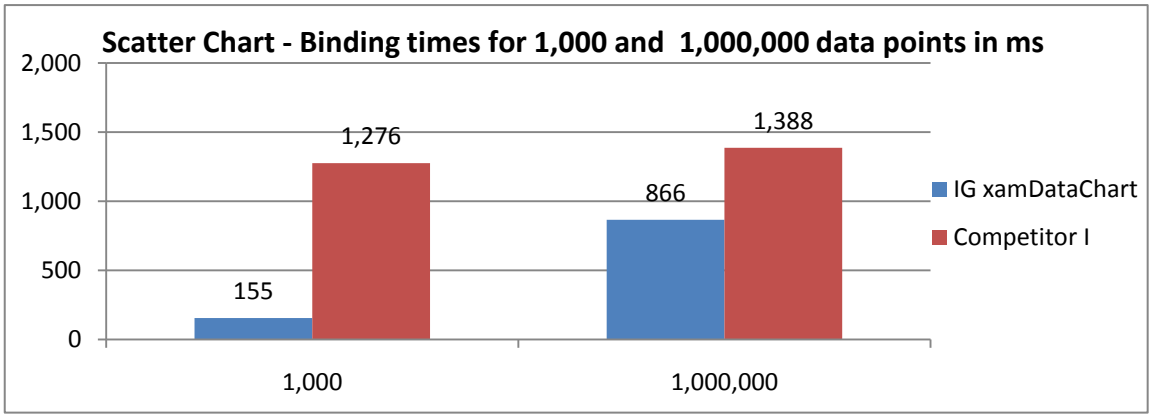


Table 3: Scatter Series chart binding

Type	Points #	Metric	IG xamDataChart™	Competitor I	
Scatter Series	1,000	Time duration in ms	155	1,276	723% Slower
		Memory Usage Before in KB	1,681	1,355	19% Less
		Memory Usage After in KB	2,033	6,687	228% More
	1,000,000	Time duration in ms	866	1,388	60% Slower
		Memory Usage Before in KB	35,180	34,274	2% Less
		Memory Usage After in KB	53,024	53,769	1% More



4.2 Zooming/Scrolling

Zooming is de facto a combination between vertical and horizontal scrolling.
Because of that and in order to keep this whitepaper brief, we will show numbers only for zooming. The statistics related to zooming performance can be used as a proxy for scrolling performance.

We use a Silverlight automation framework for zooming scenarios which sets a new scale for the chart by dragging and selecting with the mouse a rectangular region to zoom into. We calculate the time duration and memory usage between the raising of the MouseLeftButtonUp and LayoutUpdated events of the chart.

After these zooming/scrolling tests we realized because most of our competitors performance samples are enabled only with horizontal scrolling their samples are noticeably slower if both scrolling types are enabled.

Table 4: Area Series zooming

Type	Points #	Metric	IG xamDataChart™	Competitor I	
Area Series	1,000	Time duration in ms	9	208	2211 % Slower
		Memory Usage Before in KB	4,964	6,824	37 % More
		Memory Usage After in KB	5,245	4,360	16 % Less
	1,000,000	Time duration in ms	20	809	3945 % Slower
		Memory Usage Before in KB	42,770	57,534	34 % More
		Memory Usage After in KB	42,778	36,037	15 % Less

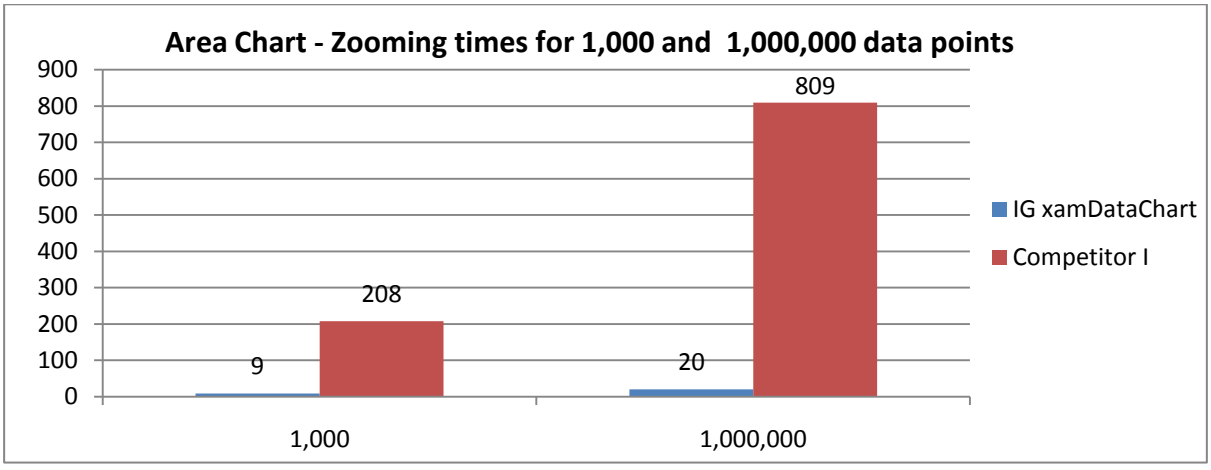


Table 5: Line Series zooming

Type	Points #	Metric	IG xamDataChart™	Competitor I	
Line Series	1,000	Time duration in ms	9	128	1322 % Slower
		Memory Usage Before in KB	5,512	6,154	11 % More
		Memory Usage After in KB	5,825	7,357	26 % More
	1,000,000	Time duration in ms	4	711	17675 % Slower
		Memory Usage Before in KB	49,497	57,405	15 % More
		Memory Usage After in KB	49,505	36,271	26 % Less

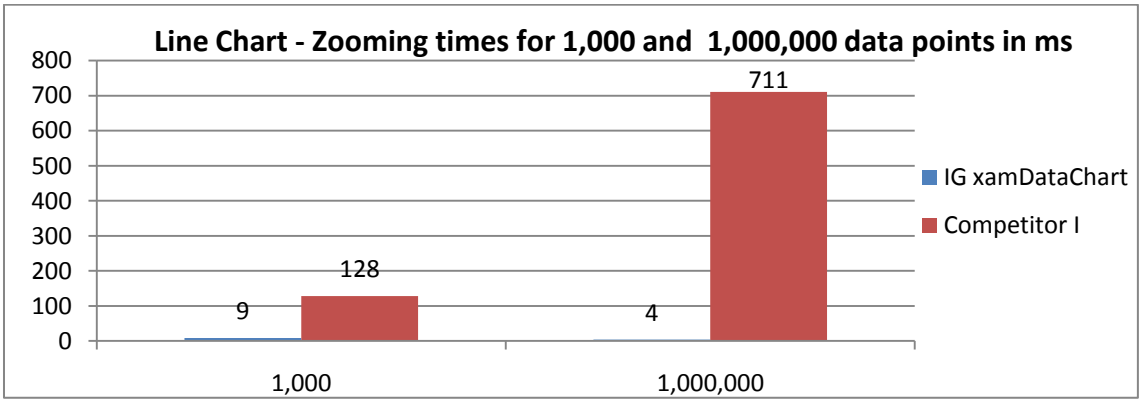
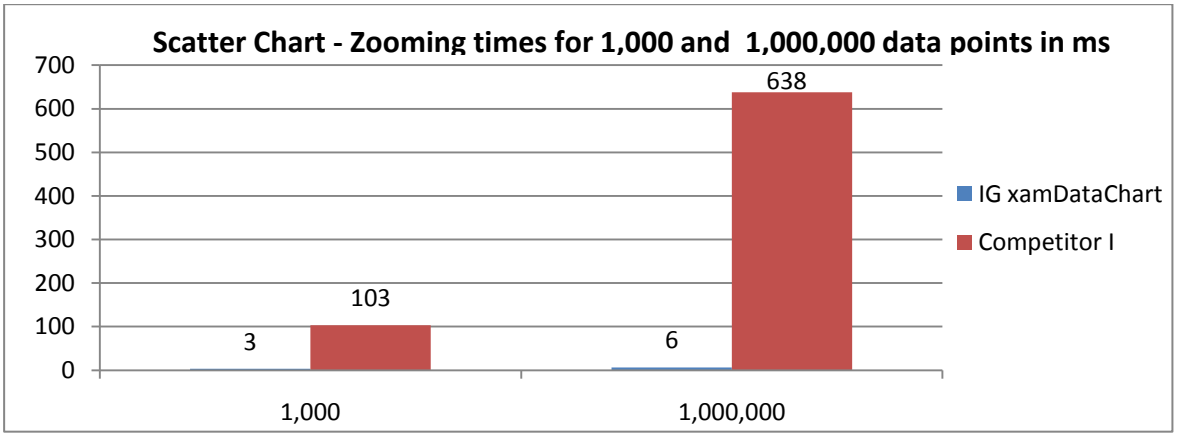


Table 6: Scatter Series zooming

Type	Points #	Metric	IG xamDataChart™	Competitor I	
Scatter Series	1,000	Time duration in ms	3	103	3333 % Slower
		Memory Usage Before in KB	6,668	6,263	6 % Less
		Memory Usage After in KB	6,710	7,146	6 % More
	1,000,000	Time duration in ms	6	638	10533 % Slower
		Memory Usage Before in KB	86,884	55,743	35 % Less
		Memory Usage After in KB	86,894	39,096	55 % Less



4.3 Update data source chart scenarios

Performance of update data source chart scenarios is important when end user needs to see change in data immediate and it's critical for the decision making.

In following section we will discuss performance of scenarios:

- Remove data points
- Add data points
- Edit data points

NB: For all "Update data source chart scenarios" we calculate the time duration and memory usage before updating the data source and after LayoutUpdated event of the chart.

4.3.1 Edit data points in a chart

Change all data points values with even indexes between 0 and 50 (Value=-Value or Y=-Y).

Table 7: Area Series – edit few data points in a chart

Type	Points #	Metric	IG xamDataChart™	Competitor I	
Area Series	1,000	Time duration in ms	13	921	6984 % Slower
		Memory Usage Before in KB	7,958	5731	27 % Less
		Memory Usage After in KB	8,036	9148	13 % More
	1,000,000	Time duration in ms	416	1245	199 % Slower
		Memory Usage Before in KB	41,074	31876	22 % Less
		Memory Usage After in KB	81,724	57053	30 % Less

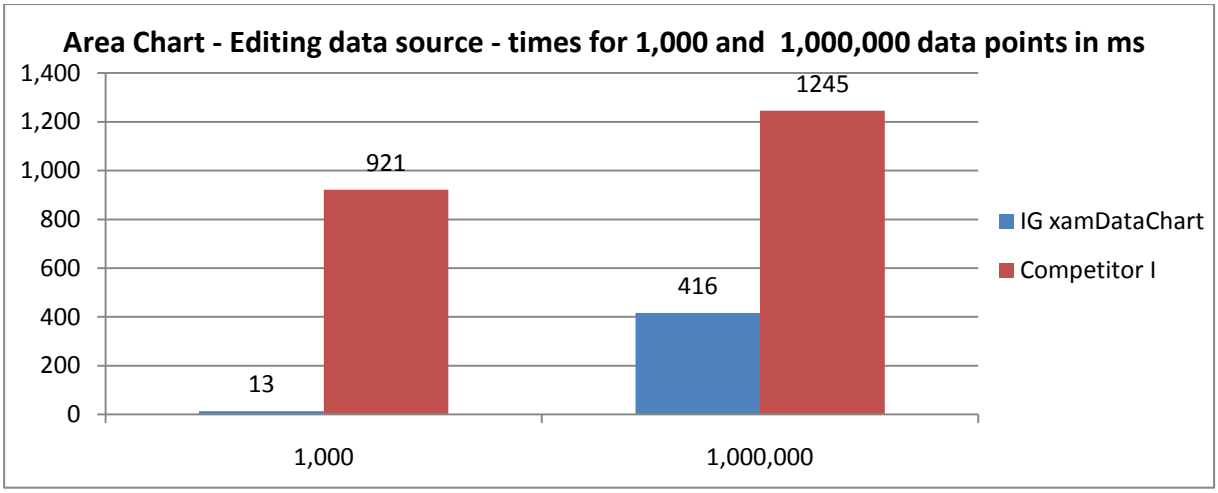


Table 8: Line Series – edit few data points in a chart

Type	Points #	Metric	IG xamDataChart™	Competitor I	
Line Series	1,000	Time duration in ms	180	752	317 % Slower
		Memory Usage Before in KB	1,793	7527	319 % More
		Memory Usage After in KB	2,065	10419	404 % More
	1,000,000	Time duration in ms	439	1133	158 % Slower
		Memory Usage Before in KB	49,258	31816	35 % Less
		Memory Usage After in KB	89,896	60409	32 % Less

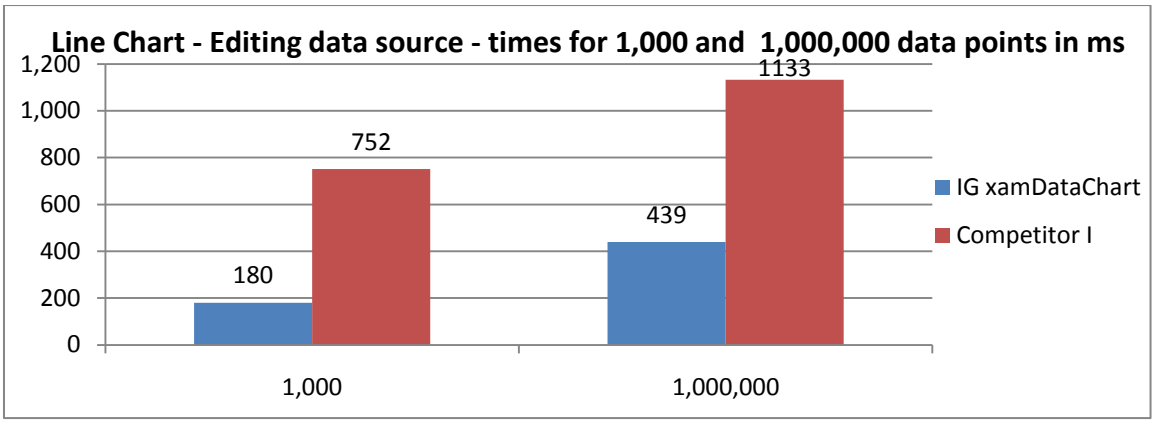
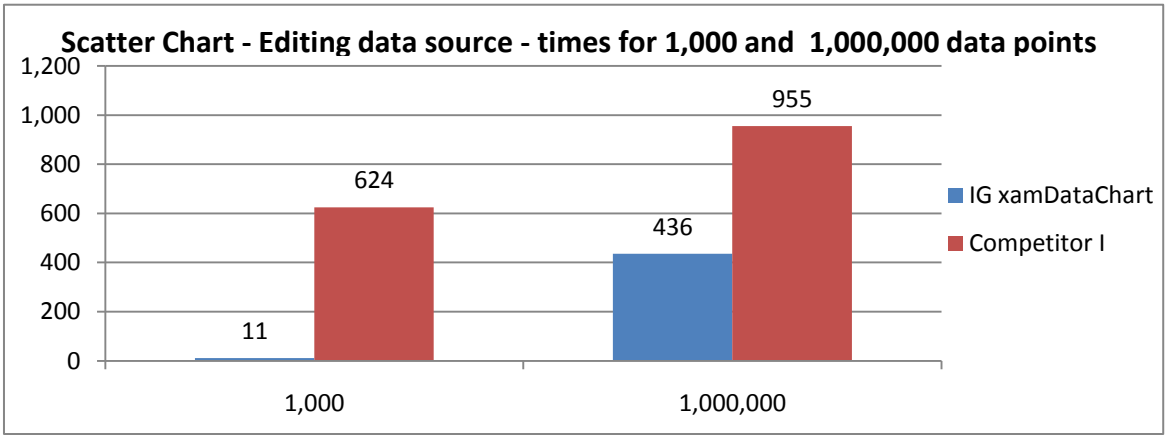


Table 9: Scatter Series – edit few data points in a chart

Type	Points #	Metric	IG xamDataChart™	Competitor I	
Scatter Series	1,000	Time duration in ms	11	624	5572 % Slower
		Memory Usage Before in KB	4,598	6037	31 % More
		Memory Usage After in KB	4,675	10512	124 % More
	1,000,000	Time duration in ms	436	955	119 % Slower
		Memory Usage Before in KB	70,110	36881	47 % Less
		Memory Usage After in KB	110,753	60440	45 % Less



4.3.2 Add data points to a chart

Insert 10 data points one by one at the beginning of the data collection.

Table 10: Area Series – add data points to a chart

Type	Points #	Metric	IG xamDataChart™	Competitor I	
Area Series	1,000	Time duration in ms	5	759	15080 % Slower
		Memory Usage Before in KB	9,041	6,069	32 % Less
		Memory Usage After in KB	9,067	8,344	7 % Less
	1,000,000	Time duration in ms	93	1,315	1313 % Slower
		Memory Usage Before in KB	42,520	36,587	13 % Less
		Memory Usage After in KB	58,155	51,411	11 % Less

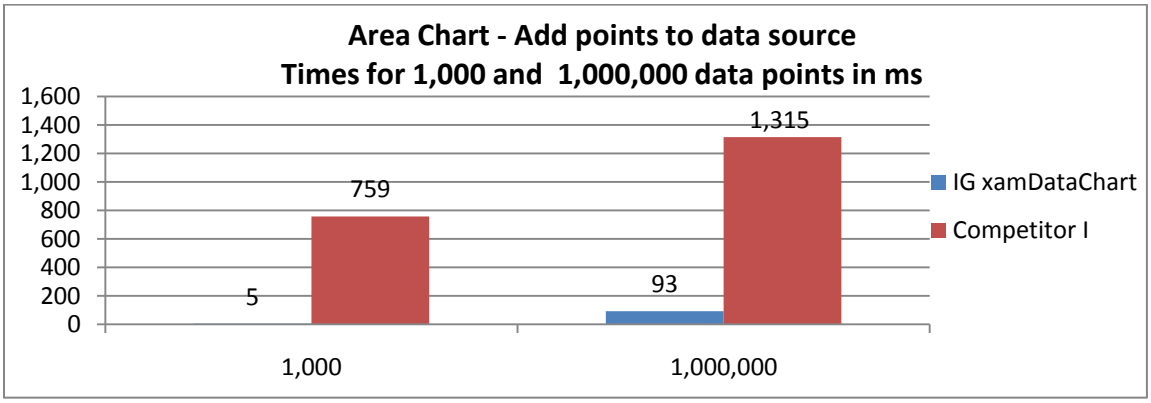


Table 11: Line Series – add data points to a chart

Type	Points #	Metric	IG xamDataChart™	Competitor I	
Line Series	1,000	Time duration in ms	5	612	12140 % Slower
		Memory Usage Before in KB	9,410	7,928	15 % Less
		Memory Usage After in KB	9,434	9,346	~Equal
	1,000,000	Time duration in ms	90	1,083	1103 % Slower
		Memory Usage Before in KB	49,258	34,436	30 % Less
		Memory Usage After in KB	64,885	59,883	7 % Less

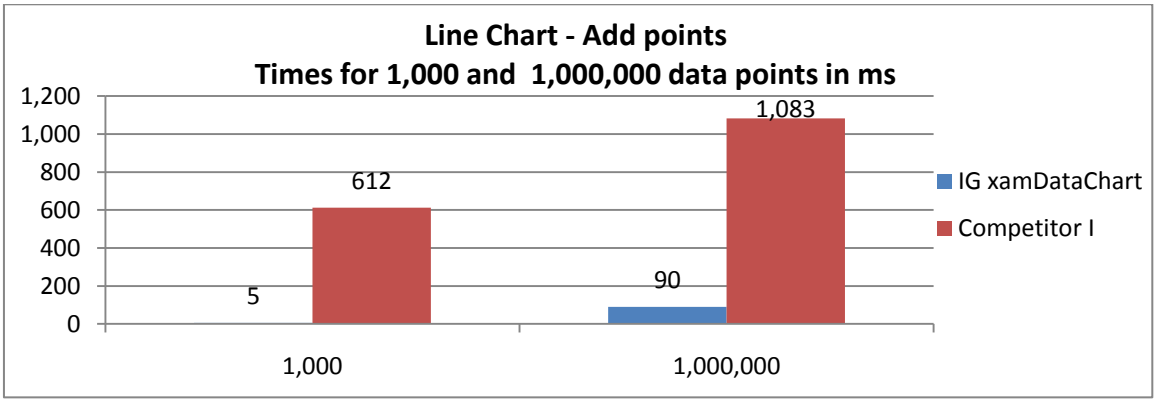
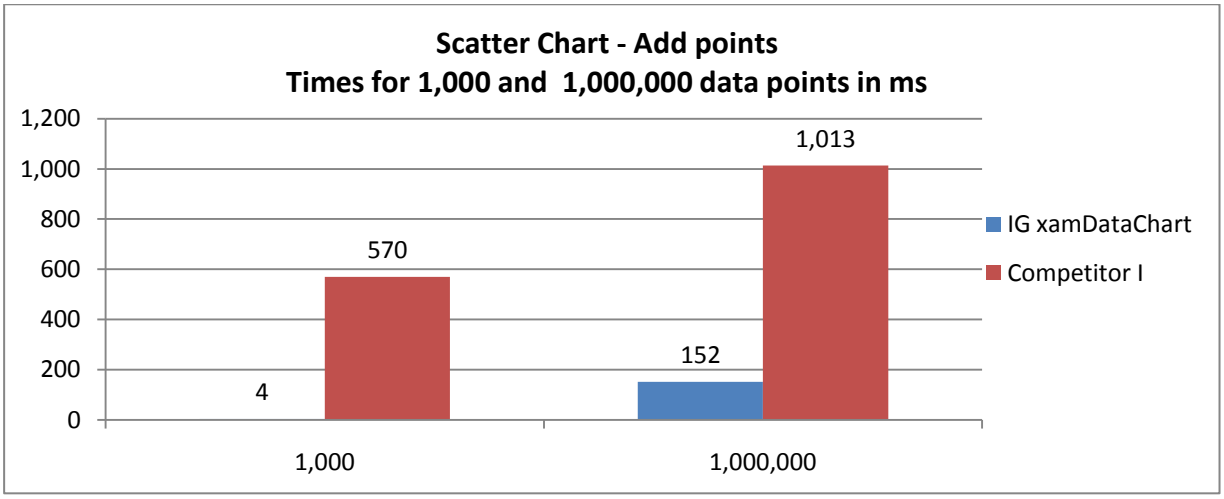


Table 12: Scatter Series – add data points to a chart

Type	Points #	Metric	IG xamDataChart™	Competitor I	
Scatter Series	1,000	Time duration in ms	4	570	14150 % Slower
		Memory Usage Before in KB	6,480	6,053	6 % Less
		Memory Usage After in KB	6,519	9,808	50 % More
	1,000,000	Time duration in ms	152	1,013	566 % Slower
		Memory Usage Before in KB	70,182	36,932	47 % Less
		Memory Usage After in KB	101,440	56,627	44 % Less



4.3.3 Remove data points from a chart

Remove all data points with even indexes between 0 and 50 (Value=-Value or Y=-Y).

Table 13: Area Series – remove data points from a chart

Type	Points #	Metric	IG xamDataChart™	Competitor I	
Area Series	1,000	Time duration in ms	7	775	10971 % Slower
		Memory Usage Before in KB	7,957	5,987	24 % Less
		Memory Usage After in KB	7,975	7,967	0 % Less
	1,000,000	Time duration in ms	199	1,323	564 % Slower
		Memory Usage Before in KB	41,163	31,530	23 % Less
		Memory Usage After in KB	41,171	62,646	52 % More

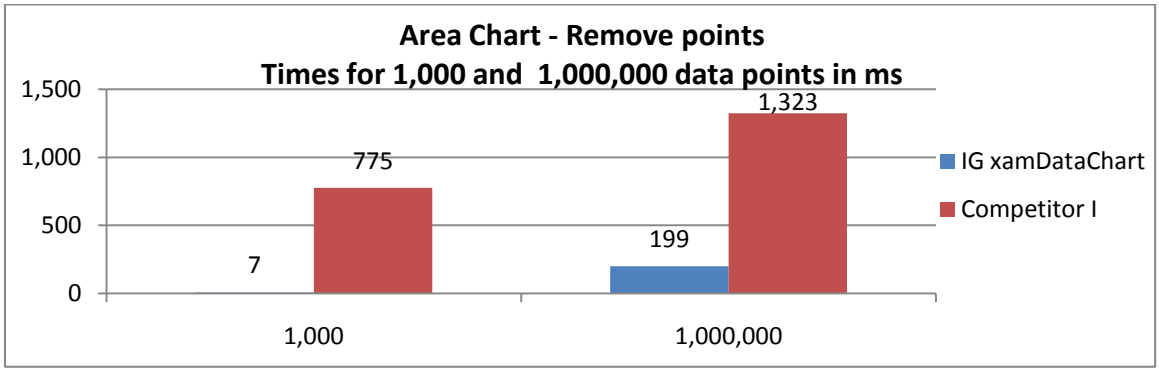


Table 14: Line Series – remove data points from a chart

Type	Points #	Metric	IG xamDataChart™	Competitor I	
Line Series	1,000	Time duration in ms	20	613	2965 % Slower
		Memory Usage Before in KB	7,984	7,925	0 % Less
		Memory Usage After in KB	8,002	9,502	18 % More
	1,000,000	Time duration in ms	203	1,116	449 % Slower
		Memory Usage Before in KB	50,175	34,499	31 % Less
		Memory Usage After in KB	50,191	59,955	19 % More

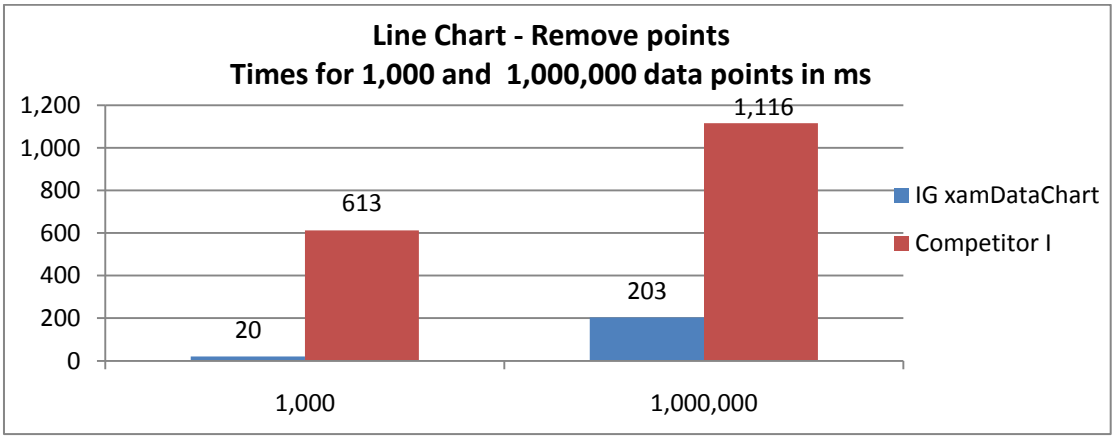
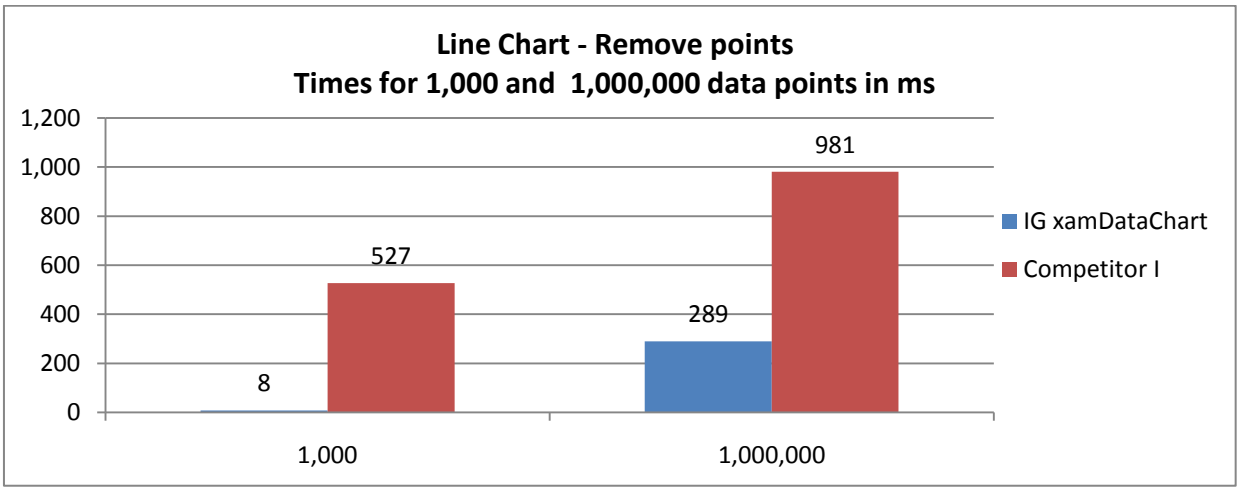


Table 15: Scatter Series – remove data points from a chart

Type	Points #	Metric	IG xamDataChart™	Competitor I	
Scatter Series	1,000	Time duration in ms	8	527	6487 % Slower
		Memory Usage Before in KB	6,455	6,046	6 % Less
		Memory Usage After in KB	6,463	9,704	50 % More
	1,000,000	Time duration in ms	289	981	239 % Slower
		Memory Usage Before in KB	86,908	36,938	57 % Less
		Memory Usage After in KB	86,916	56,586	34 % Less



4.4 Live data

When design and implement xamDataChart the primary goal is to provide a chart optimized for displaying live data with minimal performance and memory overhead. You can easily display a full screen graph and it will still scroll smoothly while running and adding new values. There's a real rendering engine behind the charts which ensures that only the changed areas of the chart are. Scrolling the chart means moving the already rendered image and only painting the newly displayed area.

During our performance testing we found that our Silverlight/WPF xamDataChart even with 1 million data points supports refresh rate from just several milliseconds (3-10 ms).
With tested competitors was impossible to achieve these results.

5 Achieving Amazing Performance

So how do we achieve such amazing performance in our xamDataChart?
The simple answer – a rock solid architecture that leverages several key techniques:

- The Data Virtualization leads to significant performance gains in xamDataChart. Using sophisticated algorithms we analyze the data and calculate how to plot fewer data points - not all data is actually displayed – it is summarized.
- Visual element virtualization for almost every user interaction operation. Visual element virtualization allows the xamDataChart to create the minimum set of Visual elements necessary to render the chart viewable area. This approach recycles Visual element objects when the viewable area changes, vastly reducing the number of object instantiation and garbage collection operations the control must perform. For large data sets with hundreds or thousands of data points these operations can have a significant impact on the performance of the application.
- Unique rendering engine. Part of this work has been to reduce the unnecessary re-renders of a chart.
- Our Data Visualization team has focused on improving the text rendering. Almost all charts have some form of text present. This text takes calculation time for layout, often requiring multiple layout passes to determine the best appearance for the chart. For example, determining the appropriate text skip or rotation of an axis label to maximize legibility can be a costly operation as multiple options need to be considered to determine the one that works best.
- Simplified Visual Tree. In the following section you can read more about the Visual Tree and why it is important to simplify it as much as possible.

6 Understanding the Visual Tree

Silverlight/WPF uses a concept called a Visual Tree to track all of the visual elements in an application. Any object that offers a visual representation (anything that derives from FrameworkElement) must be added to the Visual Tree in order to have Silverlight/WPF render it.
Visual elements can be nested, thus forming a tree-like hierarchical structure of elements that Silverlight/WPF renders to the screen. As you can imagine for all but the simplest applications the number of nodes (or Visual element elements) in the Visual Tree can grow large. The xamDataChart simplifies the Visual Tree, providing a performance advantage to applications which use it.

7 The ROI of Choosing the Infragistics xamDataChart

In order to get an idea of what the overall ROI (return on investment) is for a high performing chart in a typical line of business application, let's take to a look at the normal operations an end user might experience during the daily use of the applications you build. Imagine that a financial institution uses your application. For a trader each moment is important to take the appropriate decision. They should take these decisions based on data which is visualized in front of them.
Usually, in order to take such a decision, a financial application needs to do following operations:

- Data binding
- Zooming/Scrolling
- Remove data point
- Add data points
- Edit data points
- etc.

Let's compare the xamDataChart against "Competitor I"

Table 16: ROI of choosing xamDataChart in a Silverlight application, using a **Line Series** chart with **1 million data points**, numbers below show time duration in milliseconds

Operation	IG xamDataChart™	Competitor I	
Binding	635 ms	1,580 ms	148% Slower
Zooming	4 ms	711 ms	17675% Slower
Remove Data	203 ms	1,116 ms	449% Slower
Add Data	90 ms	1,083 ms	1103% Slower
Edit Data	439 ms	1133 ms	158% Slower
TOTAL	1,371 ms	5,623 ms	310% Slower

Using xamDataChart would help you to take the right decision faster than any other XAML chart on the market. This could lead to result in tens of thousands of dollars a day or a week and potentially millions of dollars per year in savings.

8 Summary

You can see that by employing a solid performance testing framework, you can further drive value in terms of actual saved time and perceived experience. We achieve this by using a solid architectural foundation based on a virtualized element behavior. We also follow strict best practices methodology of setting up and running tests - each test is performed on the latest releases of Infragistics and our competitor's latest service releases.
Using the xamDataChart, your applications will not only look better and be easier to use, but they will outperform anything available on the market today. This leads to higher end user satisfaction and actual dollar savings in productivity of the applications that you deploy.

To get the Infragistics xamDataChart today, download the NetAdvantage® for Silverlight Data Visualization and/or NetAdvantage® for WPF Data Visualization product at http://www.infragistics.com/downloads