Performance in the Infragistics® jQuery igGrid™

An Infragistics Whitepaper
Published June 13th, 2011
Last updated July 15th, 2012

Contents

1 Performance and User Experience

One of the most important considerations when launching a new software application in today's competitive economy revolves around a company's ability to facilitate the ultimate user experience for its customers.

While most people associate an application's user experience with the application's aesthetic design, the design is merely one aspect of the overall user experience. Performance, however, is equally important. .

At Infragistics, we take your application's user experience very seriously. We focus on helping you provide a world class visual experience, while ramping up the performance experience of our products for the ultimate benefit of your application deliverable. In this whitepaper, you will learn how we approach the architecture and testing of our Infragistics jQuery igGrid, so that we can deliver the **fastest grid on the market** for you.

The Importance of Application Performance

Why is application performance so important to the overall user experience?

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

Let's face it, your customers do not want to be, nor should they be bothered with the details of how your application was developed. The only thing your customers are concerned with is the experience they will enjoy when using your software.

Customers want an application that simply "makes sense"—an application that provides them with an intuitive user interface, and performs as they would expect. When you choose our Infragistics igGrid in our Ignite UI product package (former NetAdvantage for jQuery) you will get a control that allows you to provide that "makes sense" experience to your end users, including the best performance on the market today.

2 Exceptional Performance Best Practices

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

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

• Configuration – In this phase we configure a testing environment that simulates 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 them many times to generate our performance statistics.

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

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

3 Testing the Infragistics jQuery igGrid

The majority of Ignite UI's performance testing focuses on its igGrid.
igGrid includes a large number of features and is used in many data intensive applications. It is noted that when setting our performance goals and designing our tests, we consider many real-world use scenarios to help guide us. In fact, we talk regularly with customers who have applications that need grid controls that can handle very large volumes of data.

In testing the igGrid for performance, we include the following in our test environment:

- One physical machine (non-virtual)
- Fresh installed Microsoft Windows 7 SP1 64bit, Intel® Core™ 2 Duo CPU 2.40GHz, 4 GB of RAM.
- Windows Update functionality disabled.

In executing performance tests, code is written to ensure that test scenarios are run at least 100 times. The test scenarios range from simulating user clicks, to completing CRUD (Create, Read, Update, and Delete) operations. Each test run uses most recent builds of ours and competitor controls. To determine test results, we record time and memory consumption for each operation, and then calculate any differences.

4 Performance Results

Our test applications use the latest public bits from our grids as well as bits from our top five jQuery competitor grids. Test applications are designed to ensure that we have compared "apples-to-apples"—and simulate real world scenarios found in banks, call centers, insurance companies, etc. For comparison purposes, we use public information found on our competitors' websites.

For the purpose of this white paper, we quote results for the Infragistics jQuery igGrid™ against its two best competitors. For the purpose of anonymity, we named these competitors: "Competitor I" and "Competitor II".

Our jQuery igGrid™ has the unique feature of being both a pure client grid and MVC grid.

It is noted that none of the competitors include DOM virtualization – this means that competitive products need to create DOM elements for cells that are not in the visible area of their grid.

Having said that, it is noted that we compare our grid to non-virtualized grids, simply because competitors do not have virtualization capability.

The following are primary data display and editing grid scenarios that are critical in the overall user experience:

- Flat data binding
- Sorting (string, numeric)
- Filtering (string, numeric)

Regarding the performance results in this white paper, it is important to note that unless otherwise specified, we show test results using a data set that includes 10 columns (all main data types) and 1,000 rows of data.

Furthermore, our grid configurations are based upon the following:

• Fixed height of the grid, if there is a property similar to our "*autoAdjustHeight*" property it is set to false. In most business applications you have a fixed size where you put in your grid).

• Fixed width for each column

• Virtualization that is set to false, if something else is not mentioned

• Alternate rows (common to most of business applications)

• All scrollbars set to be visible Infragistics jQuery igGrid property:

  - "*applySortedColumnCss*" is set to false
  - "*jQueryTemplating*" is set to false
  - "*LocalSchemaTransform*" is set to false

Again, we recommend that all users read the Achieving Amazing Performance section of this white paper to understand even more about how to tune the Infragistics jQuery igGrid for best performance.

To continue, the white paper also reviews speed in terms of milliseconds, as well as memory and CPU usage.

Speed and memory are detailed per browser. While CPU usage is almost identical between Infragistics and its competitors (although the jQuery igGrid CPU usage is slightly better). Also, our hover-highlight effect is also slightly more responsive and the scrolling experience is practically the same.

jQuery grids performance comparison

It's not an easy task to compare performance of the existing HTML5 grids on the market; simply many of them are immature with just few events which are not enough to measure operations such as rendering, sorting, filtering, etc. Infragistics igGrid has more than 150 events. Because we have many properties and events, you can achieve very fine-grained control over what happens in your application built using Infragistics controls.

It seems that the most critical feature requiring a grid for good performance is **Virtualization**. You can learn more about Virtualization in the section "*5 Achieving Amazing Performance.*" Among researched grids (top 10 on the market) only few grids have **Row Virtualization** and only Infragistics igGrid has **Column Virtualization**.

Therefore we analyze grids into two main types: Virtualized and Non Virtualized.

We measured: All grids in this section are bound to local 1,000 JSON records with 10 columns loaded at a single time on the client.

- Initial Load
  - Begin – from the moment exactly before the corresponding grid control starts loading
  - End – links to the corresponding event exactly after the grid is rendered or the layout is updated
- Sorting
  - Begin – from the moment exactly before a sorting operation
  - End – exactly after the sorting operation is finished (without rendering)
- Filtering
  - Begin – from the moment exactly before a filtering operation
  - End – exactly after the filtering operation is finished (without rendering)

Browser – Internet Explorer® 9

Not virtualized grids

Table 1: Operation durations in milliseconds under Internet Explorer® 9

IE9	Data Type	Infragistics igGrid (Not Virtualized)	Competitor I		Competitor II	
Initial Load	All	774	1,327	71% Slower	1,128	45% Slower
Sorting	Numeric	805	1,448	79% Slower	2,712	236% Slower
	String	793	1,470	85% Slower	2,721	243% Slower
Filtering	Numeric	1081	1,536	42% Slower	2,618	142% Slower
	String	218	267	22% Slower	1,609	638% Slower

* All % are calculated for Infragistics igGrid to ensure that we compare apples-to-apples. Competitor I grid does not support Virtualization.

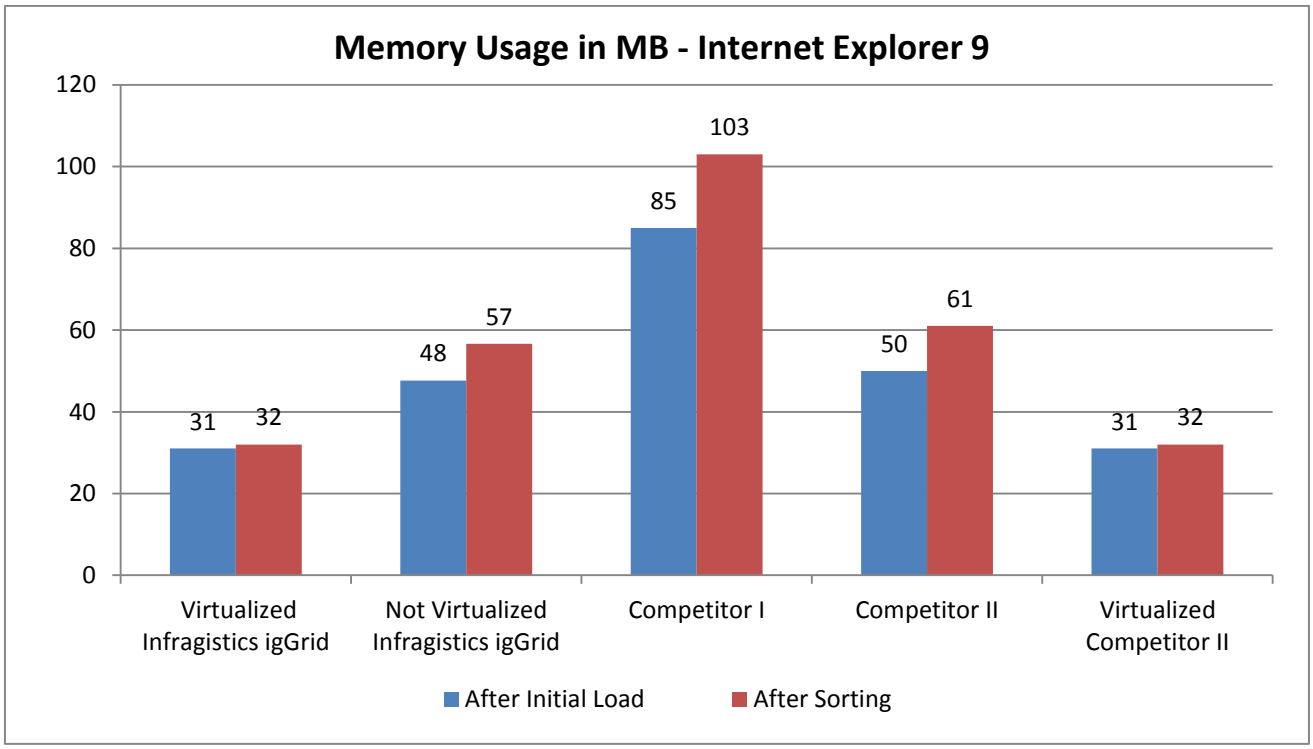
Virtualized grids

Table 2: Operation durations in milliseconds under Internet Explorer® 9

IE9	Data Type	Infragistics igGrid (Virtualized)	Competitor II (Virtualized)	
Initial Load	All	114	192	68% Slower
Sorting	Numeric	48	63	31% Slower
	String	61	63	3% Slower
Filtering	Numeric	62	64	3% Slower
	String	23	41	78% Slower

* All % are calculated for Infragistics igGrid to ensure that we compare apples-to-apples.

Competitor I grid does not support Virtualization.



Browser – Firefox® 13

Not virtualized grids

Table 3: Operation durations in milliseconds under Firefox® 13

FF13*	Data Type	Infragistics igGrid (Not Virtualized)	Competitor I		Competitor II	
Initial Load	All	652	1,581	142% Slower	1,435	120% Slower
Sorting	Numeric	699	1,257	79% Slower	983	40% Slower
	String	556	1,236	122% Slower	1,011	81% Slower
Filtering	Numeric	254	1,259	395% Slower	1,003	294% Slower
	String	74	132	78% Slower	437	490% Slower

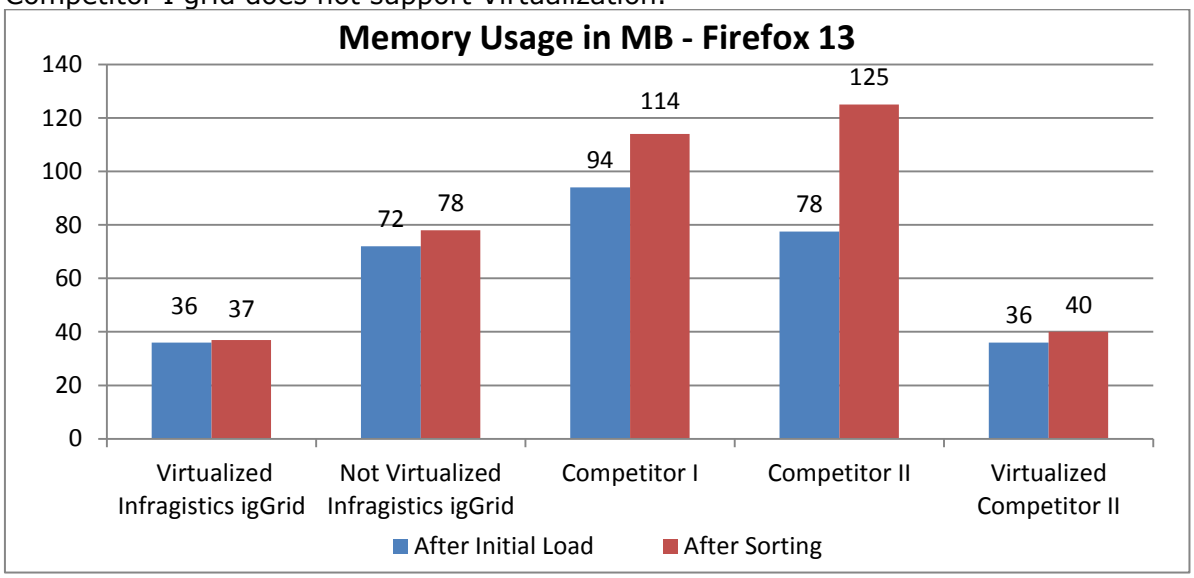
* All % are calculated for Infragistics igGrid to ensure that we compare apples-to-apples. Competitor I grid does not support Virtualization.

Virtualized grids

Table 4: Operation durations in milliseconds under Firefox® 13

FF13*	Data Type	Infragistics igGrid (Virtualized)	Competitor II (Virtualized)	
Initial Load	All	118	161	36% Slower
Sorting	Numeric	50	53	6% Slower
	String	70	72	2% Slower
Filtering	Numeric	43	67	55% Slower
	String	35	41	17% Slower

* All % are calculated for Infragistics igGrid to ensure that we compare apples-to-apples. Competitor I grid does not support Virtualization.



Browser – Chrome® 19

Not virtualized grids

Table 5: Operation durations in milliseconds under Chrome® 19

Chrome 19*	Data Type	Infragistics igGrid (Not Virtualized)	Competitor I		Competitor II	
Initial Load	All	530	1,068	101% Slower	725	36% Slower
Sorting	Numeric	547	1,118	104% Slower	914	67% Slower
	String	543	1,162	113% Slower	951	75% Slower
Filtering	Numeric	529	1,206	127% Slower	916	73% Slower
	String	62	142	129% Slower	286	361% Slower

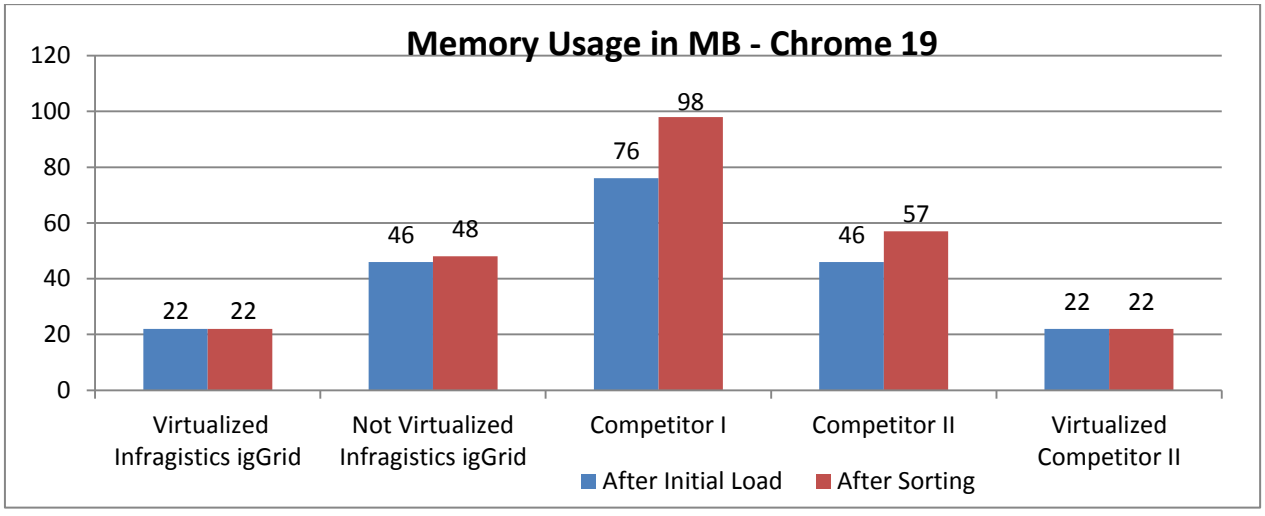
* All % are calculated for Infragistics igGrid to ensure that we compare apples-to-apples. Competitor I grid does not support Virtualization.

Virtualized grids

Table 6: Operation durations in milliseconds under Chrome® 19

Chrome 19*	Data Type	Infragistics igGrid (Virtualized)	Competitor II (Virtualized)	
Initial Load	All	57	84	47% Slower
Sorting	Numeric	47	34	27% Faster
	String	43	31	27% Faster
Filtering	Numeric	30	32	6% Slower
	String	30	31	3% Slower

* All % are calculated for Infragistics igGrid to ensure that we compare apples-to-apples. Competitor I grid does not support Virtualization.

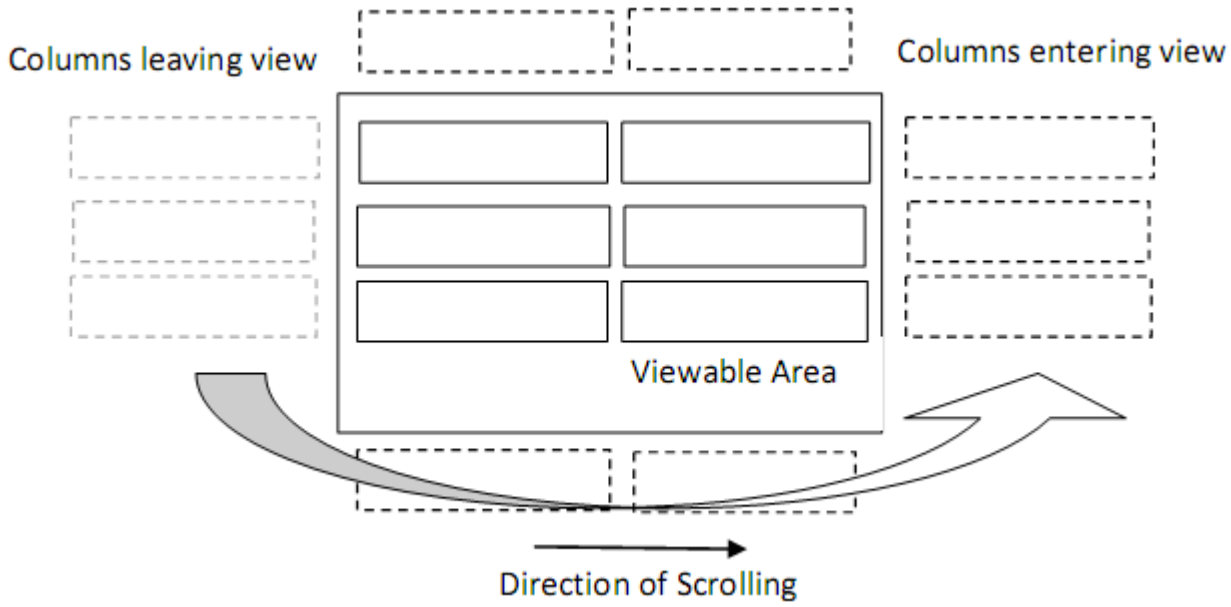


5 Achieving Amazing Performance

So how do we achieve such amazing performance in our jQuery Grid control? The simple answer is that we have a rock solid architecture that leverages DOM Virtualization for almost every user interaction operation.

DOM Virtualization allows the Infragistics jQuery igGrid to create the minimum set of HTML elements necessary to render the grids viewable area. Virtualization is possible by supporting a virtual DOM section in the grid that is constantly refreshed for data. This means that HTML elements for the entire data set are not explicitly created, instead DOM elements are reused in the grid. Refreshing data in existing DOM elements boosts performance significantly for scenarios that require large sets of data without using paging or virtual scrolling to fetch more data from the server.

In the figure below you can see how both the igGrid's DOM Virtualization infrastructure actually reuses both Row and Column HTML elements as they scroll out of the visible area of Infragistics jQuery igGrid.



6 The ROI of Choosing the Infragistics jQuery igGrid

In order to get an idea of what the overall ROI (return on investment) is for a high performing grid in a typical Web application, let's take to a look at the normal operations that an end user might experience during the daily use of the applications built across different browsers.

In fact, let's compare the igGrid against "Competitor I" and "Competitor II" measurements.

Table 7: Not virtualized grids - ROI of choosing igGrid in a Web application, using different web browsers.

	Data Type	Infragistics igGrid (Not Virtualized)	Competitor I		Competitor II	
Initial Load - IE9	All	774	1,875	142% Slower	927	19% Slower
Sorting - IE9	Numeric	805	1,100	36% Slower	2,202	173% Slower
	String	793	1,364	72% Slower	2,210	178% Slower
Filtering - IE9	Numeric	1,081	1,271	17% Slower	2,155	99% Slower
	String	218	472	116% Slower	1,584	626% Slower
Initial Load - FF13	All	652	1,910	192% Slower	657	0% Slower
Sorting - FF13	Numeric	699	824	17% Slower	783	12% Slower
	String	556	977	75% Slower	782	40% Slower
Filtering - FF13	Numeric	254	838	229% Slower	784	208% Slower
	String	74	313	322% Slower	447	504% Slower
Initial Load - Chrome 19	All	530	1,309	146% Slower	592	11% Slower
Sorting - Chrome 19	Numeric	547	795	45% Slower	680	24% Slower
	String	543	775	42% Slower	625	15% Slower
Filtering - Chrome 19	Numeric	529	815	54% Slower	611	15% Slower
	String	62	275	343% Slower	303	388% Slower
SUMMARY		8,117	14,913	83% Slower	15,342	89% Slower
AVG per all browsers		2,029	3,728	83% Slower	3,836	89% Slower

* All % are calculated for Infragistics igGrid to ensure that we compare apples-to-apples. Competitor I grid does not support Virtualization.

Table 8: Virtualized grids - ROI of choosing igGrid in a Web application, using different web browsers.

	Data Type	Infragistics igGrid (Virtualized)	Competitor II (Virtualized)	
Initial Load - IE9	All	114	192	68% Slower
Sorting - IE9	Numeric	48	63	31% Slower
	String	61	63	3% Slower
Filtering - IE9	Numeric	62	64	3% Slower
	String	23	41	78% Slower
Initial Load - FF13	All	118	161	36% Slower
Sorting - FF13	Numeric	50	53	6% Slower
	String	70	72	2% Slower
Filtering - FF13	Numeric	43	67	55% Slower
	String	35	41	17% Slower
Initial Load - Chrome 19	All	57	84	47% Slower
Sorting - Chrome 19	Numeric	47	34	27% Faster
	String	43	31	27% Faster
Filtering - Chrome 19	Numeric	30	32	6% Slower
	String	30	31	3% Slower
SUMMARY		831	1,029	23% Slower
AVG per all browsers		208	257	23% Slower

* All % are calculated for Infragistics igGrid to ensure that we compare apples-to-apples. Competitor I grid does not support Virtualization.

As you can see from the table above, the Infragistics jQuery igGrid increases your productivity between 2 and 10+ times when compared to the competition's products (depending on whether a virtualized Infragistics jQuery igGrid is used or not). This could result in tens of thousands of dollars a week saved and potentially millions of dollars per year in savings.

7 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 igGrid, 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 jQuery Grid control today, download the Ignite UI product at http://www.infragistics.com/downloads