In-memory Analytical Systems: Perspective, Trade-offs and Implementation
Executive Summary

This white paper examines the pros and cons of in-memory analytics with particular reference to TIBCO Spotfire analytics solution. It is divided into four main sections: Introduction, Trade-offs, Overview of In-memory Approaches and TIBCO Spotfire Analytics's approach. The Introduction discusses the balance act engineers encounter in developing analytical systems. Trade-offs looks at the fundamental differences between in-memory and disk-based approaches. This is followed by an analysis of the different in-memory approaches. Finally, the engineering considerations of the TIBCO Spotfire approach are explored.
Introduction

Transactional databases are just that; they are databases designed to collect transactions. The designers who build the engines (Oracle, SQL Server, DB2, etc.) that underpin these transactional systems strive to build the best engines they can but, like any engineering problem, they have to reconcile a number of conflicting constraints. For example, everyone wants the engine to process transactions as quickly as possible. However those transactions must be internally consistent (the data integrity must be retained) and the transactions must demonstrate persistence (that is, they must survive over time); ensuring both of these inevitably impacts the transaction rate. Finding the right balance between speed, integrity, persistence and the host of other factors is what makes the difference between an average database engine and a great one.

We see this balancing act in many areas of engineering. Dragsters are carefully honed for the single task of accelerating as rapidly as possible - which makes them complete disasters as family cars. Their turning circles are wider than the average parking lot and then there are the flames from the exhausts…. In the same way, transactional (OLTP – On Line Transaction Processing) systems are bad at anything other than transaction processing – analysis for example.

The problem is that the work loads are simply too different. Transactional systems often store each piece of data once and once only to help ensure data integrity, but doing so splits the data up into multiple tables with multiple joins between them. Analytical systems often have to perform huge aggregations across millions of rows so it is far more efficient to analyze just one table. However to get all the data into one table means we have to store multiple copies of each piece of data.

So are all analytical (OLAP - On Line Analytical Processing) systems the same? No. One major difference is that some hold the data on disk (disk-based systems) and others hold it in memory (in-memory systems). On the face of it, memory has an over-riding advantage over disk; it is much, much faster – something like three orders of magnitude faster. But like all engineering decisions, there is rarely one over-riding factor. It turns out that designing a good in-memory analytical system is as much a careful balancing act as designing a good transactional one.

The next section takes a look at the fundamental pros and cons of in-memory analytical systems. These factors are the same for any in-memory analytical system and are not specific to TIBCO Spotfire.
Trade-offs: In-memory vs Disk-based Approaches

Pros and cons are always comparative. If we say that a given technology is faster we must have some other technology in mind for comparison. The alternative to RAM (Random Access Memory) is disk - HDD (Hard Disk Drive).

Essentially there are four areas where comparison is significant:
- Speed
- Persistence
- Price
- Volume

**Speed:** This is definitely, without question, a pro for in-memory systems. When do you ever hear someone ask for his analytical system to run more slowly? Speed is very, very important in analysis and RAM is faster than disk. Much faster.

**Persistence:** One of the most important characteristics of an OLTP system is ‘data persistence’ - a measure of how long the data remains in existence after it has been created. RAM, of course, loses the data as soon as the power is disconnected, but the persistence requirements for OLAP are significantly different from those of OLTP. Since we typically analyze data taken from the OLTP systems, we can rely on their ability to persist data to ensure that data is never lost.

**Price:** RAM is expensive compared to disk, very expensive.

Current retail prices are in the order of:
- HDD – 1 TByte – USD$120 = 12 cents per GByte
- RAM – 1 GByte = USD$30 per GByte

**Volume:** 32 bit systems can theoretically support 4 GBytes of RAM but in practice they’re often restricted to perhaps 3.5 GBtyes. 64 bit systems have a theoretical limit of 16.3 billion Gbytes (16.3 Exabytes) of RAM - in practice there are other limitations such as the operating system.

There is no point in suggesting that disks aren’t far cheaper and far more readily accessible in large volumes. Persistence is vital in transactional systems but is not usually an issue for in-memory analytical systems. The significant advantage of in-memory systems is that, with the right engineering, they can be made much, much faster than disk-based systems.
Let’s take a look at some of the practical trade-offs that are encountered in real situations:

1. **Memory Availability and Cost vs. Real World Data Volumes for Analysis**

One question of paramount importance at this point is “How much data do the people who use in-memory analytics actually want to analyze?”

Memory is expensive and limited in volume compared to disk. But if the target users never need to analyze more than, say, 3 GBytes of data, then these are not restrictions in any practical sense. Another way to approach this question is to ask how much data fits into, say, 1GByte?

Let’s assume one table with 20 columns. Ten of the columns are text (limited to, say, 50 characters but with an average length of seven characters), five are dates, three integers and two reals. A 1.5 million row table saved as a CSV file is 236 Mbytes – about a quarter of a GByte. If we double the number of columns and rows, that’s about 1 GByte. 40 columns and 3 million rows is a sizeable chunk of data, depending on the kind of analysis you want to perform.

So one GByte of RAM may sound trivial when matched against the TBytes available on disk but for many people in many analytical situations, it is more than enough. If it isn’t enough, but 5-10 million rows will cover it, then the extra 100-200 USD for the RAM isn’t going to break the bank. However if you need to analyze TBytes of data in a single analysis, and people do, then in-memory systems are probably not for you.

2. **Central Control vs. Ad-hoc Data Exploration**

Most enterprises already have existing Business Intelligence (BI) systems. These pull copies of the transactional data from the operational systems and load it into a data warehouse (disk-based for persistence). Data is then extracted from the warehouse and moved to analytical systems.

One huge advantage of a BI system is that it allows centralized control of the data for analysis. This means that the definitions of the data can be very carefully controlled, as can the quality of the data and the access to it. BI architects consider this aspect of BI to be hugely important. Without this centralized control we have silos of data, silos of meaning and multiple versions of the truth. This can rapidly lead to analytical anarchy – ‘Excel Hell’ as it is sometimes termed.

So, centralized control is essential, but it can come at a terrible price. If we aren’t careful, it can slow the delivery of new ways of working to a crawl. How often have we heard business users complaining that “It takes IT far too long to make the changes I need.”
Don’t those guys realize I need to make this decision today?” “I’ve asked for a new analysis and all IT can talk about is its change-management system!”

Once again, we come to an engineering trade-off. Too much central control stifles innovation, rapid response and business agility. Too little unleashes the dogs of anarchy.

In-memory management, just like a chain saw, is a powerful tool. Both trees and people have limbs and a chain saw can remove both with alacrity when used thoughtlessly. In intelligent hands, in-memory analytical systems are a perfect complement to any BI system. They can take the ‘controlled’ data from the central core and allow users to perform ad hoc analysis on it. They can combine that data with other sets as and when required. If such an analysis proves worthwhile, then is the time to bring that new data under centralized control. In my opinion, in-memory analysis is a very powerful tool and represents no threat to the concept of ‘one version of the truth’ unless it is badly used.

3. Measures and Dimensions vs. Free Dimensional Analysis

In traditional analytical systems we distinguish between measures and dimensions. Measures are numerical and continuously variable. Dimensions are discontinuous and can be used to segment the measures. As a simple example, an analytical system might have two measures:

- Price Line Item
- Quantity Ordered

and three dimensions

- Time
- Customer
- Sales Person

The dimensions are typically hierarchical, so Time might have four levels:

- Day
- Month
- Quarter
- Year

We often have a very large number of values for the measures. In this example we might over ten years, receive an average of 10,000 orders per year, which might equate to 50,000 order detail records per year, half a million over 10 years.

We typically aggregate measures, so if we plot a measure against a dimension (perhaps the sum of Price Line Item against month for a single year) we are only plotting twelve values.
However, if we plotted a measure against another measure, say Quantity Ordered against Price Line Item for the same period, we would suddenly have 50,000 points on the graph.

This can be mind-numbingly slow to plot, which is why most analytical systems make the clear distinction between measures and dimensions and quietly refuse to plot measure against measure. Many users of analytical systems have simply come to accept the limitation and never even try to do it – or work around the problem. Which is odd because the inability to plot measure against measure is not a logical issue; it is merely a performance one.

With certain types of data, this ability can be immensely useful and in-memory systems have the potential to be fast enough to allow users to do this.

This brings us neatly to a differentiation between in-memory and disk-based systems that is highly significant but often overlooked – freedom from the need to pre-model.
4. Pre-Modeling of Data vs. Free Dimensional Analysis

Disk-based analytical systems force the user to pre-model the data – that is, to make many decisions about it before the analysis is performed. The need to classify data as measures and dimensions is one example of pre-modeling, defining the hierarchies within dimensions is another. Disk-based systems require this pre-modeling, because for performance reasons, they need to pre-aggregate the data before analysis can be performed. The problem is that the pre-modeling inevitably restricts the analyses that can then be carried out. It also introduces a delay between data loading and availability for analysis and means that if the hierarchies are changed, the aggregations have to be recalculated. In-memory analytical systems can be fast enough to aggregate on the fly. This allows hierarchies to be created on the fly which means that in-memory systems can be entirely freed from the need to pre-model the data for performance.

Many users simply come to accept they cannot plot measure against measure.

Figure 2 – Spotfire Analytics allows for a flexible Measure vs Measure analysis: Price by Quantity Ordered
5. Speed and User Interface Design

Stated earlier, in-memory is faster than disk based analysis. That speed can be utilized in several ways, and one of the most important turns out to be making the user interface much more responsive. This sounds like a mere convenience but it turns out to be highly significant. Since some original work done at IBM in 1984 it has been known that increasing the speed of the response of a system can significantly increase productivity. In fact, this had been intuitively understood even before 1984 but these two papers highlighted the importance of getting response times below one second. Lambert’s paper, for example, revealed that reducing the system response time from 2.22 seconds to 0.84 seconds resulted in a massive 62 percent improvement in user productivity. So, in terms of a user-interface, ‘fast’ doesn’t mean under five seconds, it means sub-second responses.

Overview of the Different In-memory Approaches

Designing an in-memory analytical system is like solving any other engineering problem. Along the way you face a large number of design decisions, each of which has pros and cons and implications. Designing the TIBCO Spotfire solution was no different.

1. Pre-aggregation

One of the greatest decisions in designing an in-memory system has to be whether or not to pre-aggregate the data (e.g., sales per month, by product group). This is very common in disk-based systems where it makes a huge speed difference so perhaps it feels intuitively right to do the same in in-memory systems.

Pre-aggregation in memory can be argued to provide a better high-level overview of the data. However it takes time and it also takes up memory to store the aggregations. We read in the data to create a single table with no pre-aggregation. This makes the data available in a simple format that is easy for the user of the system to understand. Spotfire Analytics sees two issues in pre-aggregation. First it takes time, second it enforces a level of prejudgment on the analysis that the user may be going to perform. Spotfire software can get the speed it needs (response times of 0.2 seconds) by other means. So while the Spotfire development team did look at pre-aggregation very carefully, it decided it had too many disadvantages and very few advantages.
2. Applying Heuristics

Some data for analysis arrives with meta data that describes it (data from tables in relational database systems for example) some doesn’t (CSV files). However it is possible, as data is loaded, to apply heuristics to learn about the data. If, for example, a column contains text values that look, smell and taste like dates, then it makes sense to assume that they are dates.

The huge advantage of applying heuristics is that when the user comes to analyze the data, if the software is aware that a given column is a date, the user can be presented with a range of possible date hierarchies that may be appropriate. In the same way, if a column contains numbers, and the software makes that decision during loading, then the user can be presented with an interface that allows, for example, numerical data to be filtered elegantly. The downside of taking a look at the data during loading is that it takes time to do it. The upside is that Spotfire Analytics was able to create an interface that takes intelligent account of the specific data and presents the user with a ready-to-use application with logical choices for the analysis they can perform.

3. What do you do when the volume of the data that the user wants to analyze exceeds the volume of available memory?

There are essentially three options here:

1. Crash and burn
2. Refuse to load the extra data
3. Swap to disk

Spotfire Analytics chose option three. While option two is also perfectly fair; the development team eventually decided that option three offered enough advantages to be worth implementing. The downside of option two is obvious – if you happen to need to analyze a bit more data than usual, you can’t. The downside of option three is that, when there is more data, the analysis will be slower. In addition the internal code for the in-memory system has to be more complex. It also has to keep checking (“Are we in a situation where we have to swap to disk?”) so that slows it down as well. The upside however is clear, the system doesn’t crash and it does allow you to complete the task in hand. And once you have made the decision to swap to disk, several other advantages
follow. For example, in Spotfire software’s web-based solution several users can look at the same (or different) data and simultaneously use memory on the server. If the server begins to run out of memory Spotfire Analytics makes intelligent decisions about what to do – for example, the system can page out the data that is being used infrequently (some of the users may be ‘idle’).

Ultimately disk swapping adds to the complexity of the application but Spotfire Analytics believes it is a good trade off. And, of course, the system can swap to SSD (Solid State Disk) as opposed to HDD which reduces the speed hit significantly.

4. **Should you support undo and redo?**

As users of applications such as Microsoft Office, we have all become accustomed to this functionality, so it is a feature in Spotfire Analytics. The advantage is huge – users can work in the way in which they have become accustomed. The bad part is that undo puts a very heavy load on an analytical system. You may, for example, be undoing and redoing a join to a GByte of data. Where is that data stored in the meantime? Do you store it in memory (where it is cutting down on your space for data) or on disk (assuming that your system actually allows for swapping to disk)? So undo/redo is highly desirable but it comes at a cost. Spotfire Analytics believes the benefits hugely outweigh the cost.

5. **What file size should you support?**

With the advent of 64 bit systems, the old 4 GByte limit has finally been removed. With the 3.2 release of Spotfire Analytics, the file size restriction for 64 bit systems has been removed. It now supports the file size supported by the OS; simple as that. Actually, it wasn’t quite that simple in engineering terms, but the development team believed it was important to do.

6. **Compression**

Spotfire Analytics does, of course, compress the data; indeed great care is taken to balance compression against the type of data that the user has. Sales data is generally much more compressible than scientific or financial data for example. Sales data for 10,000 products over geographic regions make up just a couple of million data values. Compressing a 100 million row data set to something much smaller is possible, while a 100 million data set of unique scientific measurements is not. Spotfire Analytics has extensive experience working with data from multiple domains. It knows that what looks good in the lab might not be that good in the real world. So the system has multiple ways of compressing the data - we look at the data as it is imported and apply the best compression algorithm.
Benefits of the Spotfire Approach

Oddly enough, Spotfire software’s overall approach to analysis didn’t start with the assumption that it had to use an in-memory data model. The goal was to improve the user’s experience in finding answers to their questions, to allow users to work with information where they don’t have to fire off complicated queries that take a long time to return the answer. The development team also wanted to free users from the old paradigms of Dimensions vs. Measures and give them more intuitive ways of working with the data. So it started with a set of design criteria that would allow it to do all of that; fulfilling those requirements ultimately led to the adoption of an in-memory model.

That essentially gave us three important capabilities that we could deliver:

1. A faster environment for end-users to work with data
2. Flexibility in working with the data to answer both anticipated and unknown questions
3. The ability to swap out to disk

1. A faster environment for end-users to work with data

It is very important that Spotfire software provide a user interface (UI) that meets Human Computer Interaction criteria – responsive, intuitive, well designed with an immediate response to user actions (within 0.2 of a second). It is well documented that productivity increases dramatically with systems that respond quickly to user actions. Compare the query, wait, result view over 30 seconds in many BI systems with one that takes only 0.2 of a second or less. In fact, users don’t come back and use systems that behave in the former way. This UI is provided automatically by Spotfire Analytics. Since the system infers what the data types and characteristics are (no modeling needed) a UI can automatically be generated so that the user can start to look at the data right away. This allows users to focus on analysis and exploration of the data instead of worrying about how to ask as few questions as possible of a slow and unresponsive system.

The thing to watch out for is - Does the system take advantage of the RAM or is it simply putting the old data architecture in memory to gain performance? We can gain speed with in-memory, but that is far from all we can do for the users.

2. Flexibility in working with the data to answer both anticipated and unknown questions

Spotfire’s in-memory implementation drops the notion of Dimensions and Measures because the system doesn’t require users to define pre calculations or worry about a user putting two measures against one another in a diagram or table, or a category vs. a measure in a scatter plot.
It allows the users to filter and re-group data on the fly. The same applies to aggregations. Spotfire Analytics doesn’t apply any modeling restrictions on the data because it is readily available and indexed as it has been read in to memory. Users are now able to work more freely with the data to find answers to their own questions – it is easier and faster leading to more insight and better decisions.

One of the other areas in which Spotfire Analytics believes its technology excels is in pulling in data from external sources and adding it to the existing data set in memory. This is nontrivial to do for many systems. Spotfire software’s design allows users to easily append new rows, and append new columns; it is optimized for both not just for one. If the user adds extra attributes the system simply generate new filters and new indices; if new rows of data are added then it update the indices.

Key question: does the system take advantage of RAM or is it simply putting the old data architecture in memory?

<table>
<thead>
<tr>
<th>Sales and Profit</th>
<th>Sales (-binary)</th>
<th>Profit (binary)</th>
</tr>
</thead>
<tbody>
<tr>
<td>subtotal</td>
<td>x = 907.75</td>
<td>1.476</td>
</tr>
<tr>
<td>subtotal</td>
<td>x = 1 462.59</td>
<td>2.672</td>
</tr>
<tr>
<td>subtotal</td>
<td>x = 1 839.25</td>
<td>7.261</td>
</tr>
<tr>
<td>subtotal</td>
<td>x = 2 428.90</td>
<td>9.733</td>
</tr>
<tr>
<td>subtotal</td>
<td>x = 2 861.97</td>
<td>10.206</td>
</tr>
<tr>
<td>subtotal</td>
<td>x = 3 359.50</td>
<td>7.083</td>
</tr>
<tr>
<td>subtotal</td>
<td>x = 3 602.25</td>
<td>9.719</td>
</tr>
<tr>
<td>subtotal</td>
<td>x = 4 368.80</td>
<td>7.271</td>
</tr>
<tr>
<td>subtotal</td>
<td>x = 4 932.75</td>
<td>127</td>
</tr>
<tr>
<td>subtotal</td>
<td>x = 5 539.50</td>
<td>125</td>
</tr>
<tr>
<td>subtotal</td>
<td>x = 6 252.35</td>
<td>72</td>
</tr>
<tr>
<td>subtotal</td>
<td>x = 6 831.90</td>
<td>135</td>
</tr>
<tr>
<td>subtotal</td>
<td>x = 7 670.75</td>
<td>90</td>
</tr>
<tr>
<td>subtotal</td>
<td>x = 8 732.50</td>
<td>153</td>
</tr>
<tr>
<td>subtotal</td>
<td>x = 9 758.25</td>
<td>05</td>
</tr>
<tr>
<td>subtotal</td>
<td>x = 10 254.25</td>
<td>220</td>
</tr>
<tr>
<td>subtotal</td>
<td>x = 11 053.75</td>
<td>56</td>
</tr>
<tr>
<td>subtotal</td>
<td>x = 11 622.75</td>
<td>97</td>
</tr>
<tr>
<td>subtotal</td>
<td>x = 12 040.25</td>
<td>134</td>
</tr>
</tbody>
</table>

Figure 4 – An example of how Spotfire Analytics disregards traditional definitions of “dimensions” and “measures”. The vertical and horizontal axes of this cross table come from Sales and Profit attributes respectively which are continuous numeric values. Traditionally these would be “measures” and could not be used on a cross table, but the Spotfire interface allows one to dynamically bin these values into groups. The values in the cross table cells are the computed average Discount for each pair of these dynamically determined “dimensions”.

In-memory Analytical Systems: Perspective, Trade-offs and Implementation
Figure 5 – Another example of how Spotfire Analytics disregards traditional definitions of “dimensions” and “measures”. The x-axis of this scatter plot is set to the names of customers, which is a categorical value. Spotfire plots are flexible in allowing dimensions or measures to drive the axes of its visuals.

Spotfire Analytics users are very happy with this capability. Many analytical systems are relatively static. You create a set of data, wait for it to aggregate and then perform the analysis. If you want to add more data, you essentially have to recompile the set for analysis. Spotfire software lets you add data on the fly.

3. The ability to swap out to disk

If you do have more data than can be held cost effectively in-memory, then Spotfire can load data on-demand into memory from a peta-byte data store. The company’s decade of experience has taught it to provide practical solutions to these problems and its On-Demand feature enable users to visually query and load portions of very large data stores into memory as needed. So, Spotfire Analytics also pages intelligently from memory to disc to make sure active users get the memory performance.
Inevitably there are limitations to in-memory systems – to claim otherwise is ridiculous in engineering terms. The question isn’t “How do we pretend that this isn’t a restriction?”, the question is “How do we deal with the limitations?” Spotfire Analytics strives to deal with them elegantly.

Summary
In-memory analytics open up new possibilities that result from the huge performance gain that in-memory offers. Those possibilities are not just in the raw speed of serving data to the user but, even more important, in simplifying models for analyzing data, provide more interactive interfaces and reducing overall solution latency.

About the Authors
Dr. Mark Whitehorn specializes in the areas of databases, data analysis, data modelling, data warehousing and business intelligence (BI). Mark is an author, academic researcher (at Cambridge and Dundee Universities) lecturer and works with national and international companies, designing databases and BI systems. In this paper, the sections that he wrote encompassed the following:

- Introduction
- Trade-offs: In-memory vs Disk-based Approaches
- Overview of the Different In-memory Approaches

Lars Bauerle is VP Product Strategy at TIBCO. In this paper, the sections that he wrote encompassed the following:

- Benefits of the Spotfire Approach
- Summary