

# How The World Works: Rows and Columns

Net Forecasts – Peter J. Sevcik  
BCR Volume 37, Number 9  
September 2007

There are two sides to application performance management: rows and columns. Let me explain.

It takes a lot of resources to deliver a screen of information to the user. These resources are typically managed by element managers and enterprise management systems. The enterprise often has a huge investment in these tools. Furthermore, the tools generally match the system architecture and organization chart. There are management systems for the network, servers, database, storage, etc.

This is what is referred to as silo management. There are good reasons to manage each silo independently: The technologies are very different. For example, management issues, resource planning and problem diagnostics are very different. You wouldn't want the network expert trying to figure out why one server with 10 users seems to be much slower than the other server with 100 users.

Smart enterprises know that they can't rely on each silo operating completely independently, so they implement "end-to-end" silo management with some enterprise management system. These systems provide an integrated view of all the silos on a single screen. There are even nice color-coded dashboards that give management a feeling of comfort when all the dots are green.

The performance thesis of this approach is that as long as you did proper capacity planning for each silo and there are no bottlenecks in the operational systems, then application performance should be good enough. The definition of "good enough" was determined by capacity planning for a projected user load with occasional stress testing of each silo.

Now let us take a completely different view of the system. This is the view as seen by the actual user's traffic. A user session is made up of a flow of packets from the client machine through some of the elements of each silo and back again. This is often referred to as the horizontal cut or flow management view of the particular

elements actually serving the user at a point in time.

Pinging a server from a desktop is the simplest form of the flow view. It requires that a small part of the desktop software is operational, with connectivity through a limited set of routers and interconnecting circuits, and finally that a small part of the server software is operating. Such a test is certainly not an indication of the health of the complete network or that the application on the server is operating. Of course the application user will need the ping path (or a comparable one) to continue operating, plus a lot more on both ends of the path, in order to have a productive application interaction.

## The Rows And Column Table

Let us start to build a table of rows and columns in our mind – like a grid. I ask you to visualize this table only because actually drawing such a table is very complex and unique for each application/enterprise/user/task combination. The silos are columns and a single user task is rows. I am limiting this to a single user task – the time from when the user hits "Enter" to when the screen refreshes so that the user can proceed to the next task.

In the 1970s, the table had just one row and one column, so it was a single cell. This is the world of terminal-host applications, and it is still operating in many areas. Each of your interactions with an ATM machine to get some cash is a series of tasks that are delivered in a single-cell performance grid.

The column is the mainframe server. The network to the ATM machine is typically a dedicated circuit with known latency and the ATM machine is a dumb terminal directly under control of the mainframe. Technically, there are two more columns, one for the circuits and another for all the terminals. But you can think of these columns as shaded. They are required for availability tracking but not for response time tracking.

In this view, each user entry is a single packet from the machine and a single packet in return. So a task is implemented as a single turn (round-trip). One host and one turn make a simple single cell table.

In the 1980s the table got more complicated with the advent of client-server computing. Now there are three columns, since we must account for the client, network, and server contributions to response time. But most client-server interactions on behalf of a user entry were still just one turn. The client pre-processed the entry into a request to the server, and the server replied with a single answer after much thinking or look-up. So we now have a table with three columns and one row to serve a single user task. (Yes, there were some more complicated client-server software implementations that would have added more rows to finish the user task, but these were the exception to the rule.)

In the 1990s, the Web arrived. A key to the success of the browser was that it used simple standardized presentation protocols upon which Web pages could be delivered. The trade-off was that to deliver a task – i.e., paint a Web page – required many turns.

In 1995, I was asked by an enterprise to study the impacts of Web traffic on their network. I profiled the tasks (Web pages) of many business-to-business sites that were used at the enterprise. The result was that the typical business page required 20 turns and 50,000 bytes. So the table now had 3 columns and 20 rows. Incidentally, this was considered a staggering demand on the enterprise network in terms of both the fast latency that would be required to deliver 20 round-trips and the bandwidth required to send 50,000 bytes, all within a target time of 6 seconds.

In mid 2007, I again profiled what I consider the touchstone of business-to-business websites – the Keynote Business 40 (KB40) – using Compuware's Application Vantage. The average number of turns has mushroomed to 88. Turns per task are consistently increasing at 13 percent compound annual growth, with no end in sight. Of course, there is a wide range here, from Google's speedy 7 turns (including DNS lookup) to the content-rich sites like CNN, USA Today, and the Wall Street Journal, each of which

weighs in at more than 200 turns. So the table just grew to 88 rows.

But there is more to the story.

As I described in last July's column, there is an explosion in the number of systems involved in delivering content to the new-age browser that I called the dynamic client. Furthermore, we have many new technologies in the datacenter since the mid 1990s. The datacenter now houses:

- Access system (firewalls, intrusion detection)
- LAN (many layers of Gigabit Ethernet switches)
- Load balancers (server directors, TCP shaping, SSL offload)
- Web servers
- Application servers
- Database servers
- SAN (storage area network)
- Storage

However, many of the servers are in fact operating within virtual machines. So there is the added complexity of:

- Virtual machines
- VM resource pools

Let us not forget the:

- Client (any kind of machine running any kind of browser)
- Network (private or Internet)

It is easy to define at least a dozen silos, all with their own management system. So the table is now 12 columns with 88 rows. The one in your head may be larger. That means there are 1,056 cells that must all deliver their little part of the puzzle just right in order to deliver the page to the user in 6 seconds. Now do you understand why you have a headache?

### **Scale – Big And Small Pictures**

Remember that each column represents a class of device for which there is a silo management system. In fact, each column has many discrete devices.

For example, there may be hundreds of servers in just one datacenter. Again the user's view is not of all the servers but the few that support a single task. Amazingly, the median number of servers involved in each Web page for the KB40 is 8. (Again this number has a wide range, from 1 for Google and a whopping 23 for USA Today.) This means that there are actually 8 different sub-columns for servers in the typical business web page. Of course, the 88 rows do not have a value in each of these sub-columns.

There is also a much larger view of rows: one for each turn from all the users and all the pages they loaded in an hour. Now the table has hundreds of columns and millions of rows.

Imagine this huge table as a tapestry. Each IT manager has a different aggregate view as they look down many columns and rows at a time. Comprehensive dashboards try to shrink the big table into something one can understand on a single screen.

But each end user is also looking at the same tapestry with a microscope that is focused on "only" 1,000 stitches that matter to him just now as he waits for the page to load. Furthermore, each user's 1,000 stitches are not even adjacent in the big tapestry. Is it no wonder that managers and users don't agree on how well an application is performing?

### **Time – The Other 1,000:1 Problem**

Let us assume that an enterprise wants to deliver the 12-column-by-88-row set of events within 6 seconds. We can start by assuming that the network component of each turn will take 60 msec (the critical user population is within this round-trip time or RTT). That means the WAN just consumed 5,280 msec for all of its entries in the table (88x60).

Now let us assign a very fast 0.1 msec to each of the seven datacenter components that operate as dedicated hardware. That takes up 61.6 msec (88x7x0.1). Finally, we "plan" that each of the four general computers running software (client, Web server, application server, DBMS) completes each of their cell entries in 1.5 msec, which adds up to 528 msec (88x4x1.5=528). The grand total for all entries in the single user task table is 5,869.6 msec (5,280+61.6+528) or just under 6 seconds.

This explains why a server manager can be claiming 1 msec response time for the application while the user sees 6 seconds, or a thousand-fold difference in the perception of time.

Of course in the real world, things are not so uniform. Each entry in the table is not exactly as planned above. In fact, the distribution of values in each column shows a lot of variability. Each task gets some values that are momentarily much higher than planned. That is why the actual measurements of many tasks across many users vary significantly and the distribution has a long tail.

The effect of the variability in the cells is that if a system were planned with the above timing budgets it is likely that the 6 second objective will actually become the best achievable response time. The response time mean of many task measurements will be significantly longer.

Small shifts in a column can make large shifts in the result if multiplied 88 times. The bottom line is that in order to ensure very good performance in Apdex terms (Apdex is sensitive to the long tail), then each column in the table must be managed to very exacting standards. Many tools do not track this, nor do systems have the controls necessary to deliver performance to such precision.

### **APM In A World Of Complexity**

Your application delivery infrastructure will not be as described here. It may be simpler or more complex. Surely not all rows have entries in all columns. For example, each turn does not have to go as far as the SAN. However, on the dynamic grid of 12 columns and 88 rows, each set of cells has a legitimate entry somewhere, depending on what the application or user is trying to accomplish at that moment--making things even more complex.

Application performance management that relies on either just the columns view or just the rows view is clearly not going to succeed. Even flow-based measurements that count traffic volumes by application-user pair or time per turn (typically a TCP transaction) are just looking at each row individually. Only task-based real-user measurements show the full time it takes to deliver a task.

The most important point for managers to remember is that the single-value world of a single-cell table is three decades old. Furthermore, any statement from a silo manager that his part is working fine or that even all the silos are working fine is a necessary but insufficient answer. APM vendors must expand their view to include all of the cells in the big table and then report on performance as seen by each user's roving microscope. These are challenges that some management vendors are starting to understand. Let us hope solutions are not far behind.

#### Companies Mentioned

Apdex ([www.apdex.org](http://www.apdex.org))  
CNN ([www.cnn.com](http://www.cnn.com))  
Compuware ([www.compuware.com](http://www.compuware.com))  
Google ([www.google.com](http://www.google.com))  
Keynote Systems ([www.keynote.com](http://www.keynote.com))  
USA Today ([www.usatoday.com](http://www.usatoday.com))  
Wall Street Journal ([www.wsj.com](http://www.wsj.com))

*Peter Sevcik is president of NetForecast and is a leading authority on Internet traffic, performance and technology. Peter has contributed to the design of more than 100 networks, including the Internet, and holds the patent on application response-time prediction. He can be reached at [peter@netforecast.com](mailto:peter@netforecast.com).*

NetForecast is an internationally recognized engineering consulting company that analyzes and improves data, voice, and video application performance. We help enterprises align application performance with business needs using a process based on the Apdex standard. NetForecast also advises technology vendors about customer requirements, technology issues, and the business value of application delivery products and services. Visit our library of educational reports and articles about performance engineering.  
**[www.netforecast.com](http://www.netforecast.com)**

