Predicting Application Performance Before Moving to the Cloud

Moving servers and applications from an existing data centre into the cloud (or a new data centre) is a project that comes with a long list of risks and unknowns.

How applications will perform – from a user's perspective – is always an unknown when systems are moved to an alternative data centre.

This presentation discusses a process and technique that measures the exact behaviour of applications.

Early analysis of the exact behaviour of applications can remove the "performance risk" and can reveal:

- Which applications/servers can be moved with no (or little) expected performance impact?
- Which applications/servers will have an expected performance impact – and how much?
- Exactly why are applications predicted to have a large performance impact?
- Workable remediation options that can be proposed and implemented.
- Remediation options that can be implemented and tested well in advance of the physical server moves.

A similar traffic analysis can be used to determine which servers should be moved as groups.

What Affects Application Performance

Network propagation delay is one of the main and often overlooked variables that can have a large performance impact. If new servers are further away from users, then the minimum network round trip time (RTT) to each server is increased. This will affect individual application transactions as well as "user functions" that involve many transactions, possibly to many servers.

Efficient applications (with a minimal number of transactions to back-end servers) can probably be moved with little performance impact.

However, inefficient "chatty" applications (that perform many transactions) can be severely impacted when the minimum per-server RTT is increased and each transaction time is correspondingly increased.

It is very common for application teams to not fully appreciate how their applications actually work "under the covers" and at the network level.

Discussed here is a mechanism that was used to evaluate key applications and user functions to:

- Predict the user impact of proposed server moves.
- Understand exactly why there would be an impact.
- Confidently provide remediation options for applications.
- Provide a general assessment of application efficiency.

This presentation can be downloaded (as a PDF) from:
http://www.networkdetective.com.au/downloads

Terminology

Term	Definition
Server Transaction	A single paired request and response message from a user's workstation (the *client*) to a server. Is at least one round trip (loop) - but may also involve multiple loops for network delivery of the request and response.
User Function	All the activity between a user clicking on a button (or similar event) and the eventual rendering of a new display on the user's screen. It may involve a large number of server transactions, to a variety of different servers. Several server transactions may run concurrently.

Example "User Functions" for different industries:

Insurance	Airline	Bank	Real Estate
• Launch "Teller" application. • Logon to application. • List all accounts. • Perform a cash withdrawal. • Perform a cheque deposit. • Transfer between accounts. • Foreign currency transaction	• Launch "Check-In" app. • Check-in one passenger. • Check-in multiple passengers. • Add a bag to booking. • List upgrade prices. • List alternate flights. • Print list of passengers.	• Launch "Teller" application. • Logon to application. • List all accounts. • Perform a cash withdrawal. • Perform a cheque deposit. • Transfer between accounts. • Foreign currency transaction.	• Create new property for sale. • Create new tenant record. • Add new tenant to property. • Process a rental payment. • Print payment receipt. • List available properties. • Print monthly statement.

Timing of Server Transactions

There are four major components that make up the timing of individual server transactions:

Time Category	Definition	Display Colour
Client time	Time spent internally to prepare the next server request	Grey
Request message transfer time	Time between the first and last packets of the request. (Zero for single packet requests.)	Green
Server time	Time the server takes to process the request and prepare the response. Measured as time of last request packet to first server response packet. Includes one RTT when captures are taken at the user end.	Yellow Orange
Response message transfer time	Time between the first and last packets of the response. (Zero for single packet responses.)	Blue

Note:
Application overheads, such as user authentication, are treated as server transactions within the overall user function.

Likewise, overheads such as TCP 3-way handshakes and SSL handshakes are also included as “transactions” within the user functions (since they also add loop delays that can be “felt” by the user).

Method Used to Measure Applications

Experienced users were chosen to demonstrate various user functions for each application.

Users were asked to start significant user transactions at given times so that each function's start time and approximate duration could be noted. For example, "press enter when the time is at 00 or 30 seconds". This helps to visually separate network traffic related to each user function.

Network traffic conveying user functions was captured, either on or close to the user's workstation, using Wireshark. Australian software, "NetData Pro," was then used to identify all the server transactions involved in the user functions and measure the four major components of response times.

NetData calculates the minimum round trip time (designated as "loop delay") to each server – as well as the number of transactions and other TCP loops involved for each server. Each user function can usually be characterised and documented with a single chart. The charts also display the various transaction types, showing any network or application inefficiencies that can form the basis for proposed remediation ahead of the server moves.

When there are concurrent TCP connections to one or more servers, NetData chooses a "critical path" through the transactions to count total loop delay (so that round trips that occur in parallel are not double counted).

The expected change of overall timing of the user function is then calculated by informing NetData of the proposed new RTT for each server. Each server can have a different "new RTT". The calculated overall performance impact is displayed in red.

Example Outputs from the Analysis

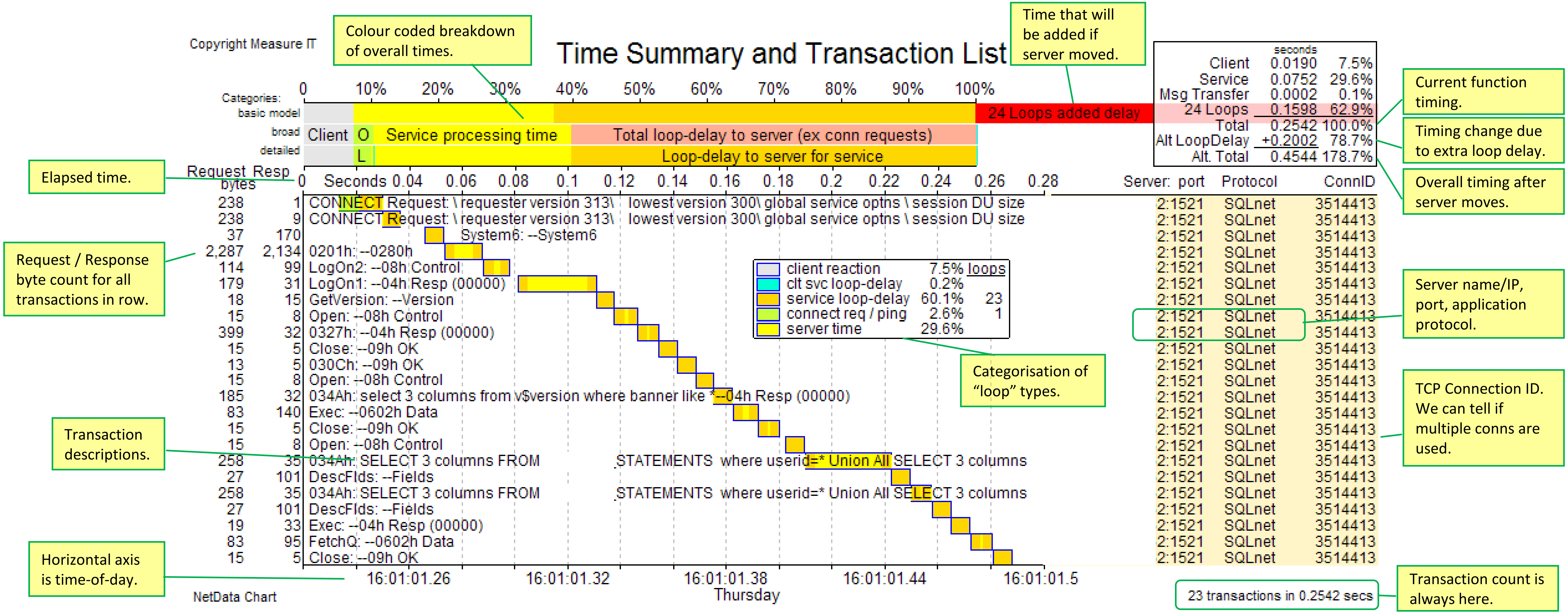
The following slides show some example “user function” summary charts from a real-life project.

Benefits of these charts:

- Show the expected performance impact to each user function.
- Show the exact reasons for the performance impact.
- Explaining it all to an application team is made easier and clearer by presenting charts rather than just text.
- The application team may learn things about their application behaviour that they hadn’t previously known.

How to Interpret the Charts

We'll begin with a relatively simple chart – which shows the 23 Oracle database transactions that were invoked for a particular end-user task that took about a quarter of a second to complete. In this case, there is only one SQL transaction per row.
Colours are used to differentiate client/server processing times, total network RTT and network time to deliver packet data.
In this case, the original 254 ms function will become 454 ms due to the added 200 ms of loop delay.



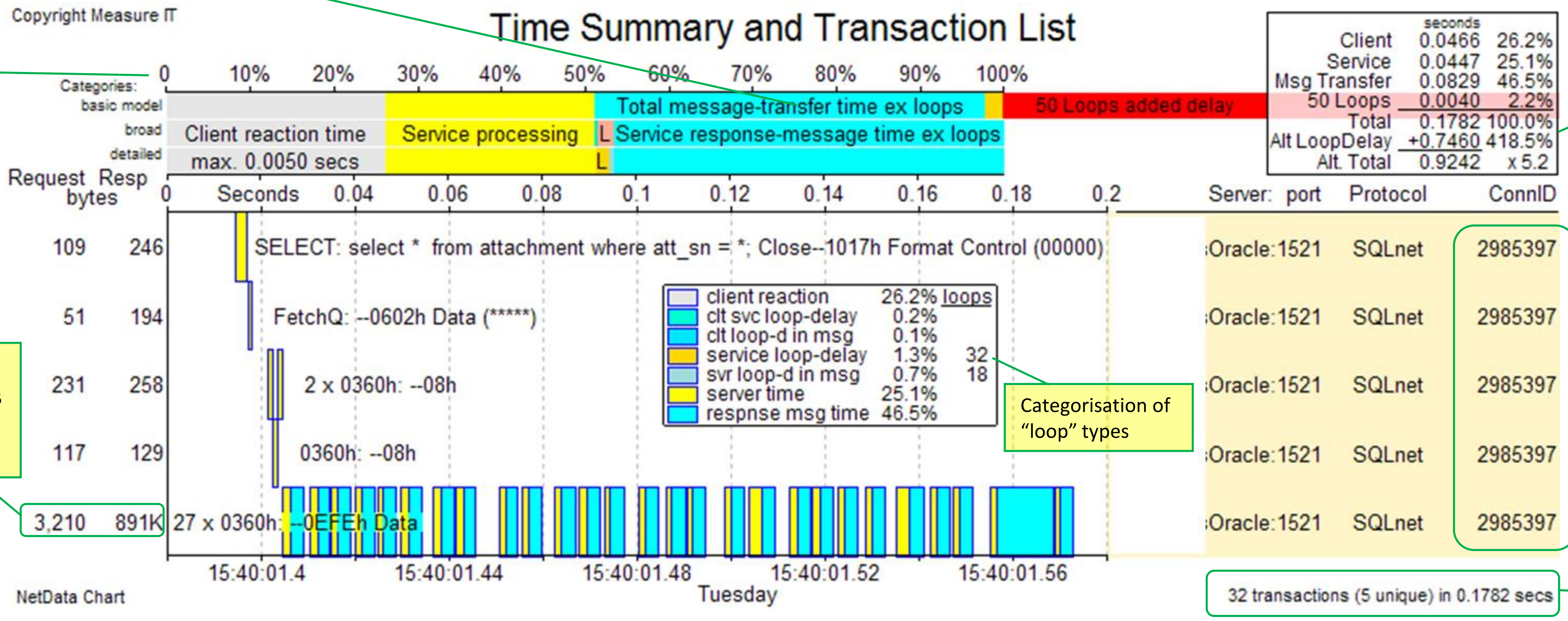
An Oracle Application

In this example, there is still only one TCP connection to a single Oracle database server. There are 32 transactions that took 178 ms to complete. In this case, each row contains multiple SQL transactions of a particular type.

The extra loop-delay would add 746 ms to the 178 ms function – to become 924 ms. Perhaps not noticeable to a human?
Could the time be reduced by combining the 27 SQL requests into one? We also need to ensure the new environment can cater for the large data transfers.

Colour coded breakup of various times. There's a lot of network time (blue) due to the large responses.

Elapsed time.



This is always the most critical table to examine.

Only one TCP connection.

27 similar SQL transactions totalled 3 KB of request data and 891 KB of response data.

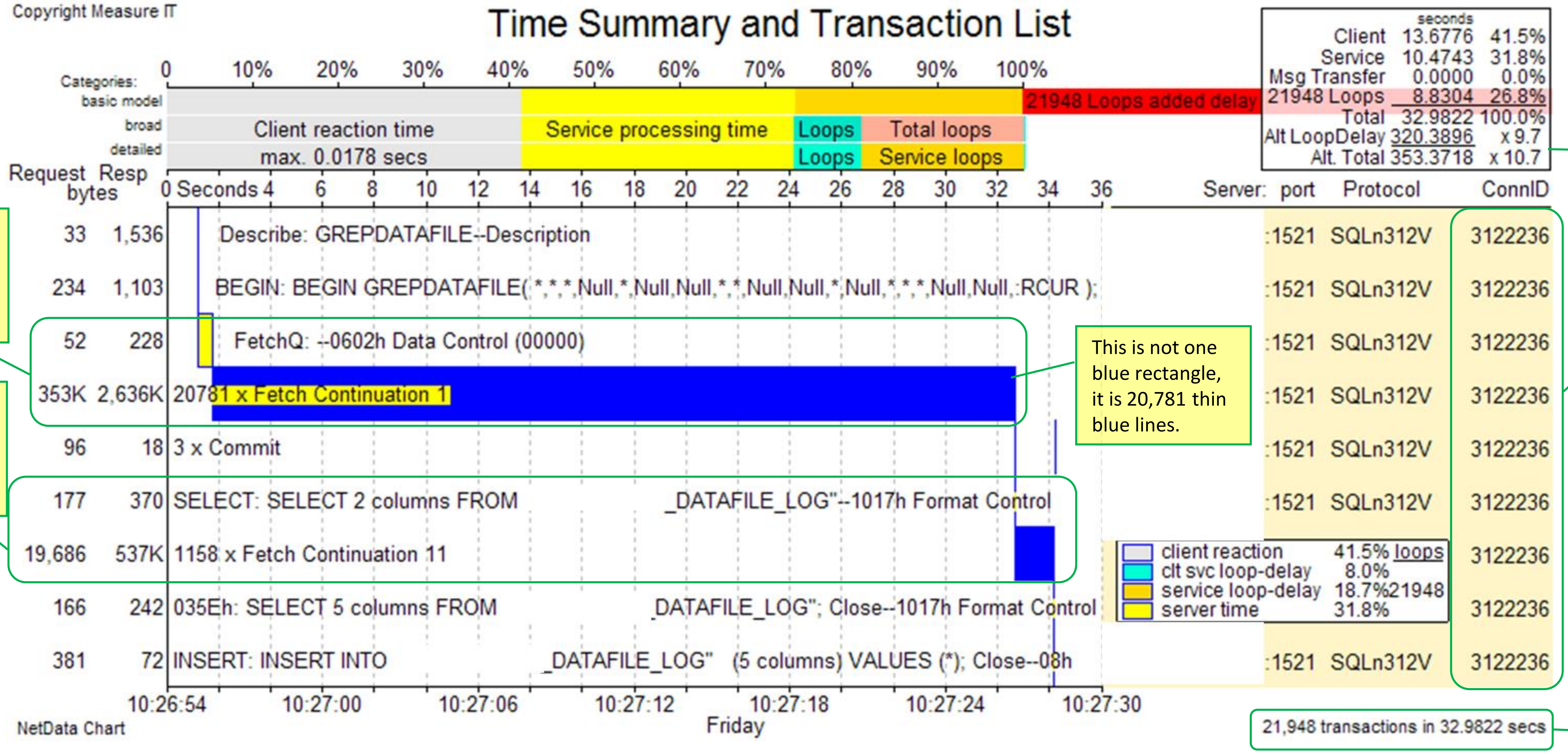
Categorisation of "loop" types

Transaction count is always here.

Inefficient Oracle Application

In this example, there is still only one TCP connection to a single Oracle database server. However, there are 21,948 transactions that took 33 seconds to complete. Retrieving thousands of rows of data either one or eleven at a time is "expensive" in both client and server processing - but extremely expensive in network round trip times. This current 33 second task would become 6 minutes.

Simple application coding changes (or driver settings) could easily eliminate 21,939 round trips and use the network more efficiently.



20,782 rows of data (2.6 MB) were retrieved one at a time, resulting in 20,782 loops.

12,749 rows of data (0.5 MB) were retrieved 11 at a time, resulting in 1,159 loops.

This is not one blue rectangle, it is 20,781 thin blue lines.

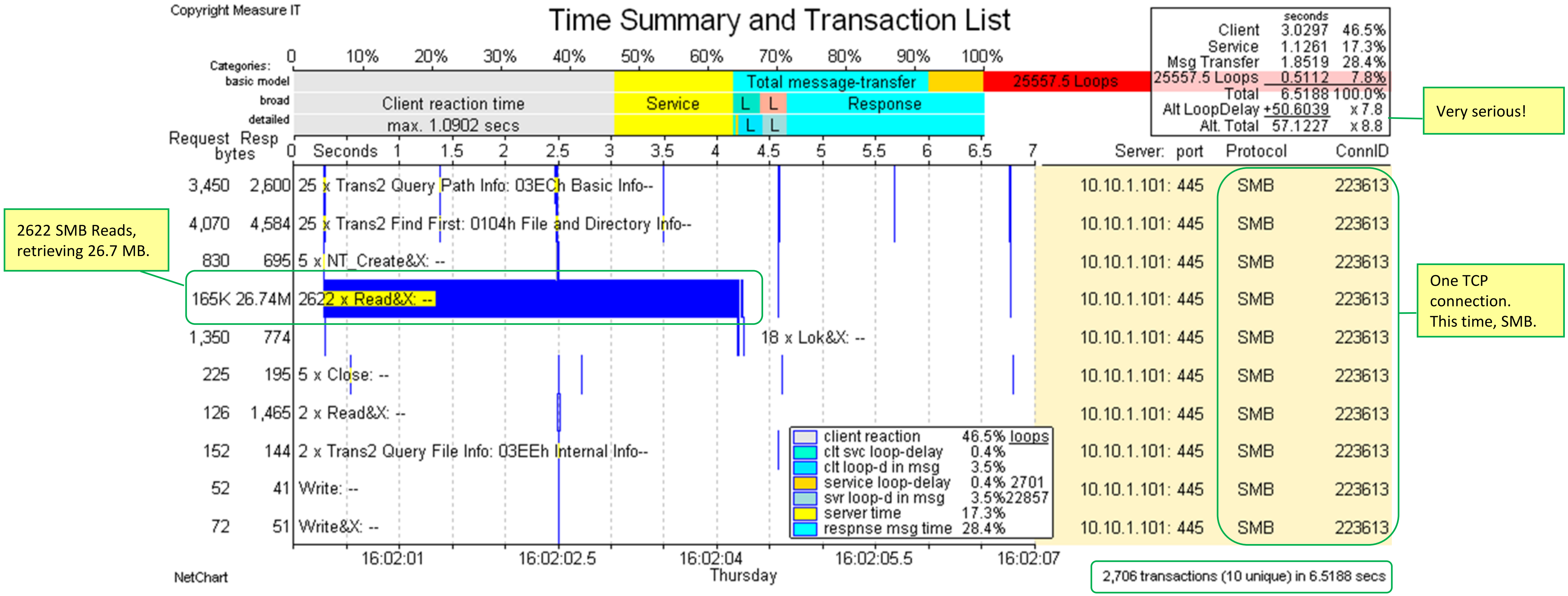
This is much more serious!

Still only one TCP connection.

Transaction count is always here.

SMB File Server

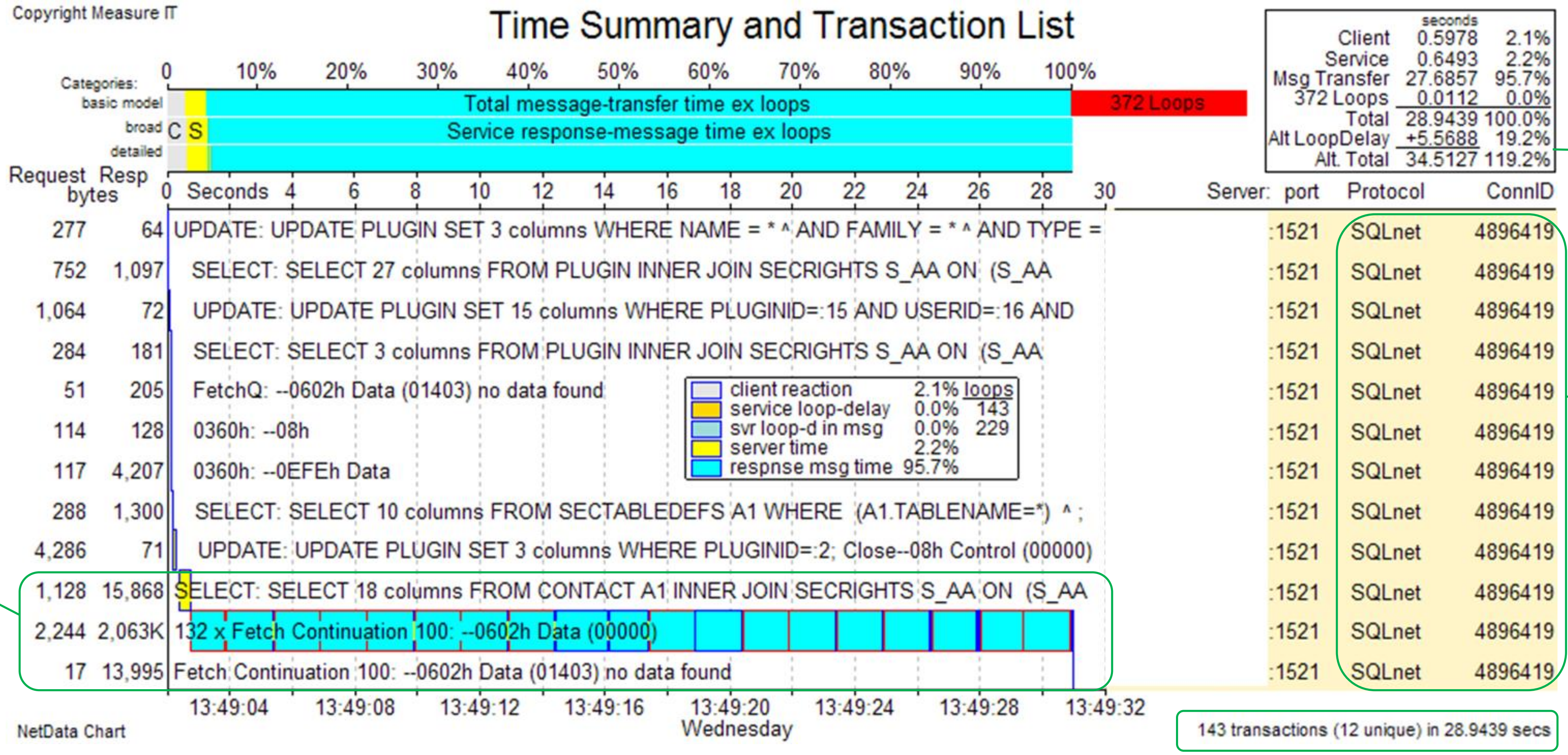
Here there is one TCP connection to a single SMB file server.
A total of 26.7 MB of data is downloaded via 2,622 Read requests (only 10 KB each) taking 4 seconds for overall completion.
This 6.5 second overall task would become 57 seconds without remediation.
Notice the large amount of grey "client" time, where the user PC has to handle all those SMB requests. It also takes 1.8 seconds of network time to transfer 26.7 MB. Could users live with this function taking nearly a minute?



Oracle Server – Network Constraint

A total of 2 MB of data is downloaded from an Oracle server via 134 requests (about 16 KB each). 95% of the overall time here was network data transfer time (blue). 28 seconds to transfer 2 MB is around 72 KB/s or 0.5 Mbps.
This 29 sec task would only become 35 secs after moving. However, network transfer speed is also an issue here, not just RTT.

Therefore, we also need to ensure that our connection to the cloud (or new DC) is provisioned with enough capacity.
It could still be worth a conversation with the application developer to retrieve larger chunks of data and reduce the 134 loops.



13,400 rows of data (2 MB) were retrieved 100 at a time, resulting in 134 loops.

An extra 5.5 secs may be OK.

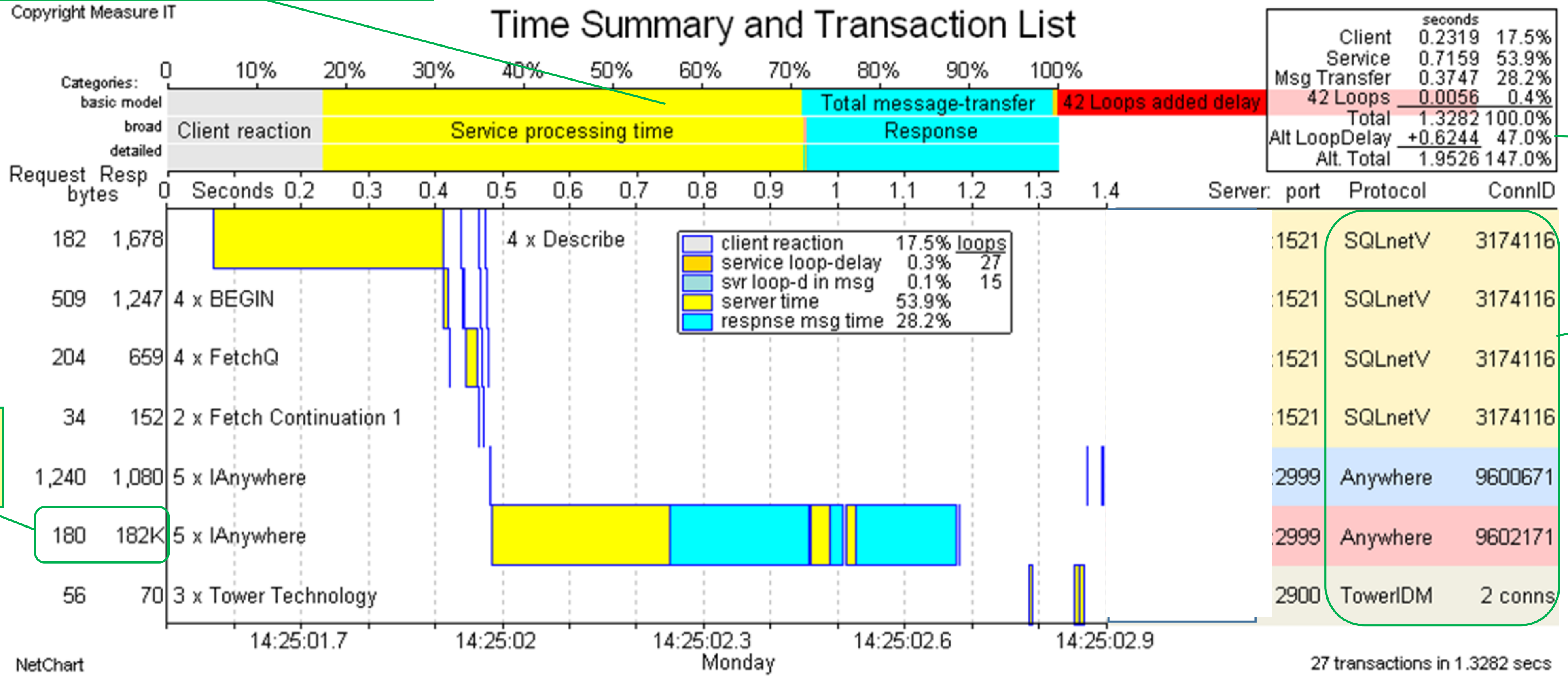
One TCP connection to an Oracle server.

Three Different Server Types

In this example, the overall function takes 1.3 seconds.
With 42 round trips, it would have just 0.6 seconds added to it (everything else being equal).

Users may not notice the extra 625 ms. However, given that server and network transfer times dominate here, we need to ensure that our new servers are capable and that our connection to the cloud (or new DC) is provisioned with enough capacity.

Colour coded breakup of times. There's a lot of server (yellow) and network time (blue) due to the large responses.



An extra 0.6 secs may be OK.

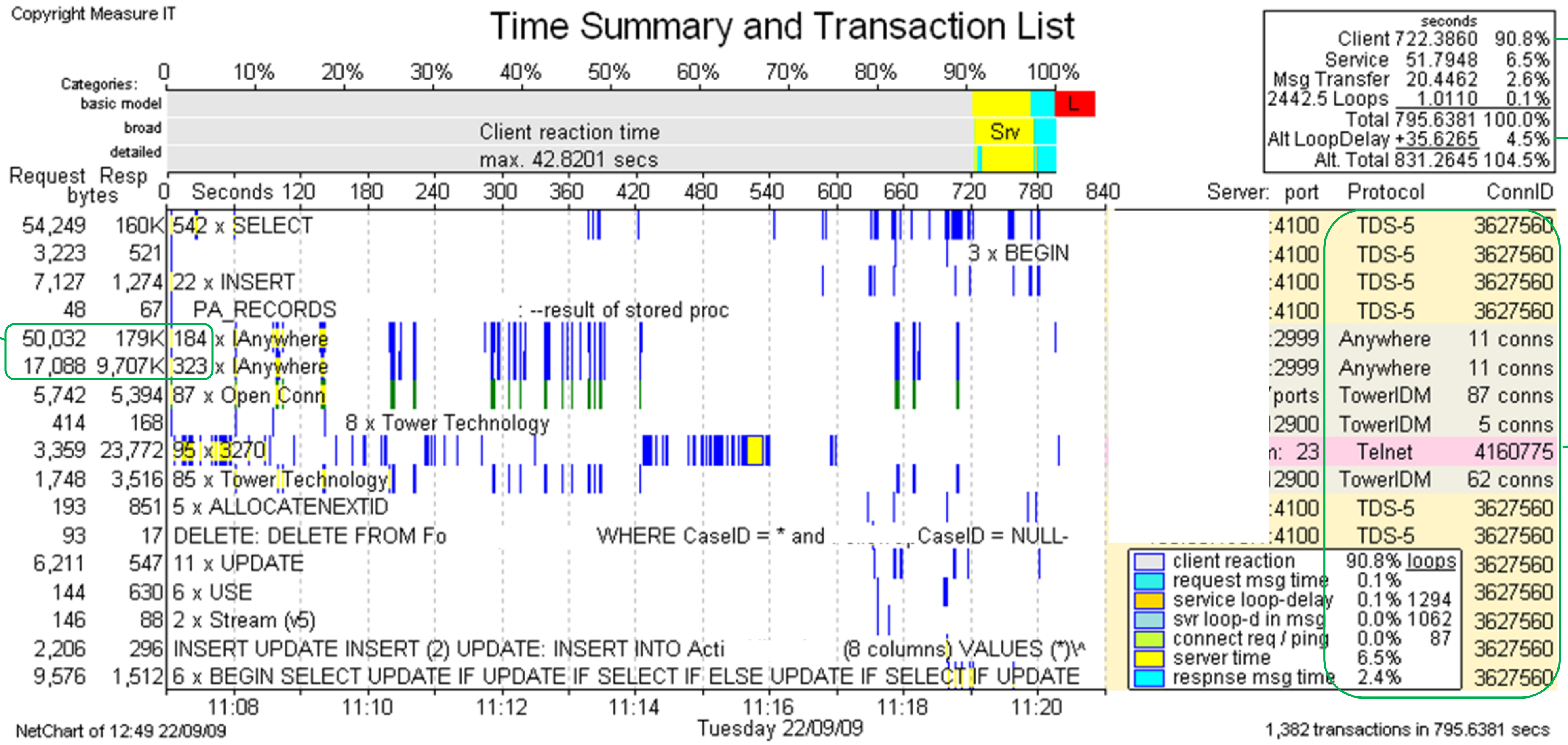
8 TCP conns to 3 different servers.

5 requests returning 182 KB.

Four Different Protocol Types

In this example, the overall function takes 796 seconds. It has a mix of protocols, including 95 TN3270 requests to a mainframe. With 2442 round trips, it would have 35 seconds added to it. A 4.5% increase may be acceptable for this function.

Of interest here is that 90% of the time for this function is grey, meaning that it is internal processing (no network activity). The place to start looking for performance improvements is within the application code on the user's PC.



Large time of no network activity.

An extra 35 secs (4.5%) may be OK.

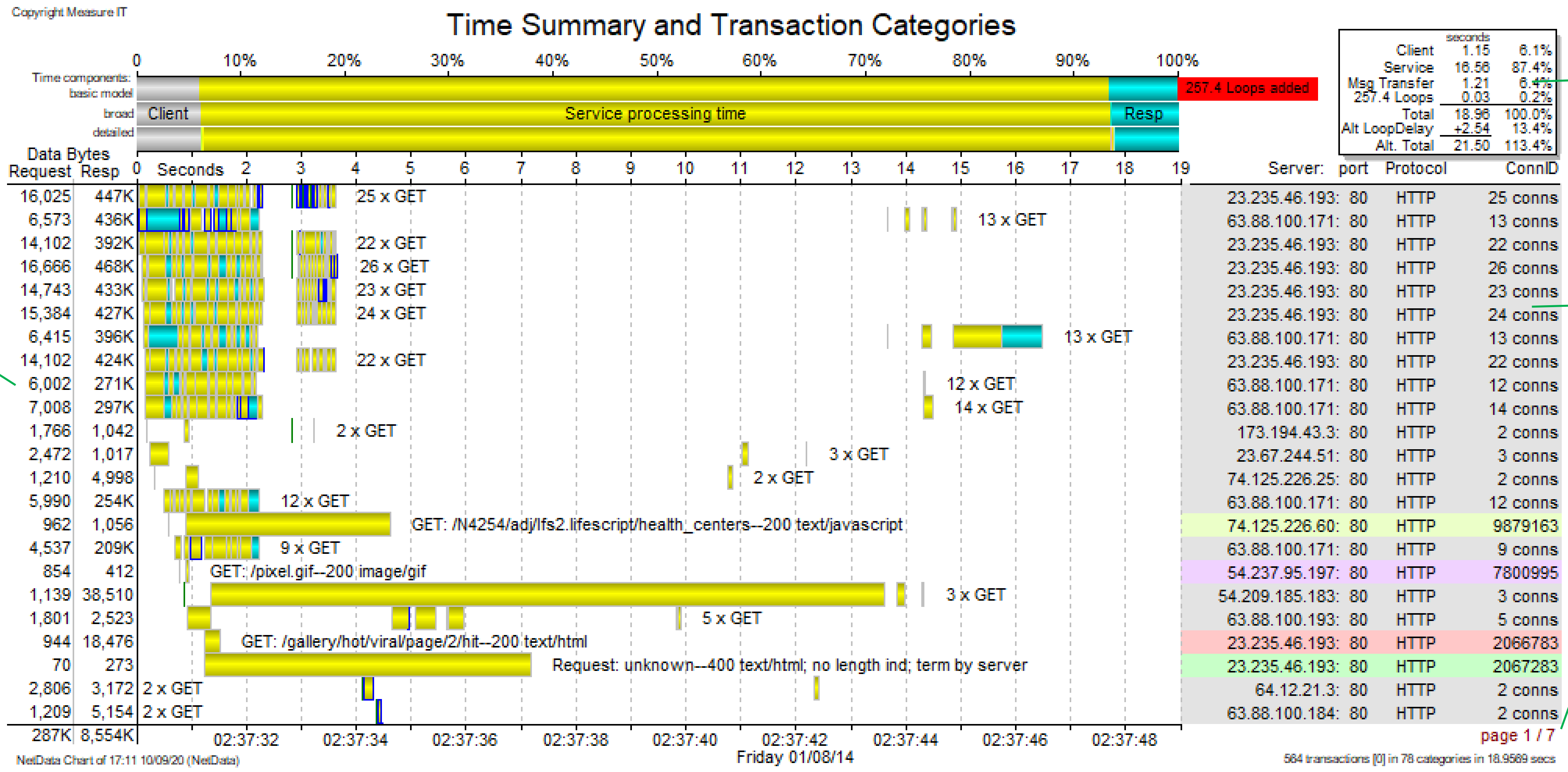
507 requests returning 10 MB.

181 conns to 4 different servers, incl mainframe.

HTTP Protocol

This is an example showing HTTP transactions. With 257 round trips, the servers being 10 ms away would add 2.5 seconds to this 19 second set of transactions (a 13.4% increase). 87% of the overall time is "Server (yellow)" with 6.4% being "Msg Transfer (blue)" – which is time taken to deliver the packets of the large responses.

It is worth noting that this chart was produced with NetData Lite. Further, that this is just page 1 of a 7 page scrollable chart.



Request	Resp	Transactions	Server	port	Protocol	ConnID
16,025	447K	25 x GET	23.235.46.193	80	HTTP	25 conns
6,573	436K	13 x GET	63.88.100.171	80	HTTP	13 conns
14,102	392K	22 x GET	23.235.46.193	80	HTTP	22 conns
16,666	468K	26 x GET	23.235.46.193	80	HTTP	26 conns
14,743	433K	23 x GET	23.235.46.193	80	HTTP	23 conns
15,384	427K	24 x GET	23.235.46.193	80	HTTP	24 conns
6,415	396K	13 x GET	63.88.100.171	80	HTTP	13 conns
14,102	424K	22 x GET	23.235.46.193	80	HTTP	22 conns
6,002	271K	12 x GET	63.88.100.171	80	HTTP	12 conns
7,008	297K	14 x GET	63.88.100.171	80	HTTP	14 conns
1,766	1,042	2 x GET	173.194.43.3	80	HTTP	2 conns
2,472	1,017	3 x GET	23.67.244.51	80	HTTP	3 conns
1,210	4,998	2 x GET	74.125.226.25	80	HTTP	2 conns
5,990	254K	12 x GET	63.88.100.171	80	HTTP	12 conns
962	1,056	GET: /N4254/adj/lfs2.lifescript/health_centers--200 text/javascript	74.125.226.60	80	HTTP	9879163
4,537	209K	9 x GET	63.88.100.171	80	HTTP	9 conns
854	412	GET: /pixel.gif--200 image/gif	54.237.95.197	80	HTTP	7800995
1,139	38,510	3 x GET	54.209.185.183	80	HTTP	3 conns
1,801	2,523	5 x GET	63.88.100.193	80	HTTP	5 conns
944	18,476	GET: /gallery/hot/viral/page/2/hit--200 text/html	23.235.46.193	80	HTTP	2066783
70	273	Request: unknown--400 text/html; no length ind; term by server	23.235.46.193	80	HTTP	2067283
2,806	3,172	2 x GET	64.12.21.3	80	HTTP	2 conns
1,209	5,154	2 x GET	63.88.100.184	80	HTTP	2 conns
287K	8,554K					

Mostly server time but some packet delivery time.

287 KB of requests returning 8.5 MB of responses.

Many TCP connections to many servers.

This is page 1 of 7 overall pages.

How to Produce a Waterfall Chart

To produce a chart like the ones in this presentation, first produce a Packet Timing chart in NetData. When loading data, packets must also be loaded in order to display the timing breakdown table at the top right of the Waterfall chart.

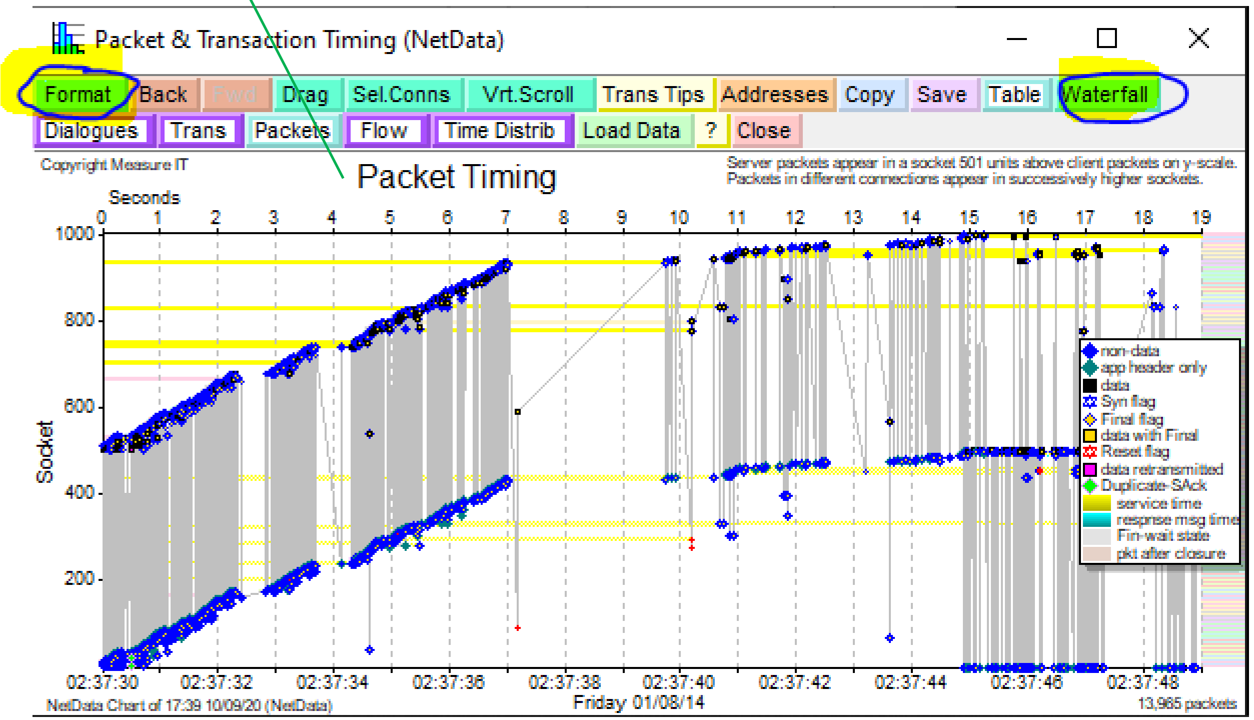
The “Waterfall” button will toggle the chart style. However, more options are available via the “Format” button. The yellow highlights show the settings used to produce the chart on the previous slide.

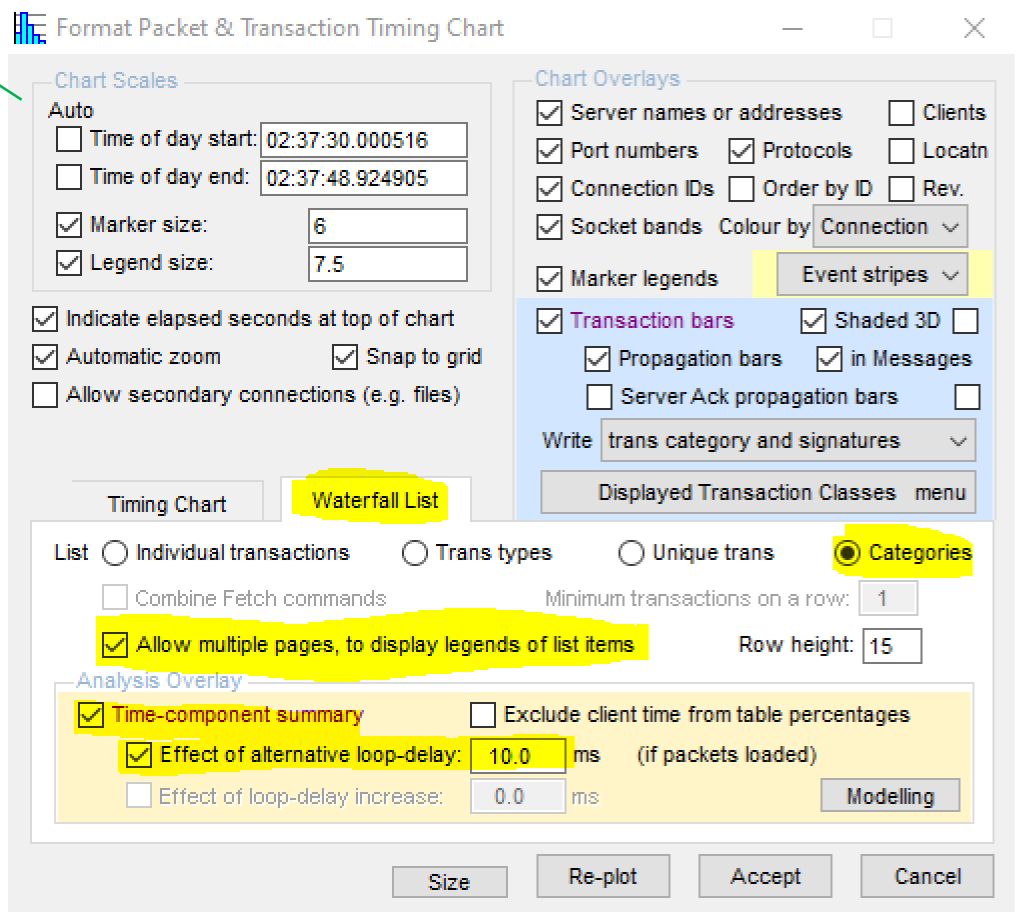
First produce a “Packet Timing” chart – which will look something like this.

To get the time summary table, packets must be loaded.

The “Format” button in the top left of the Timing chart will show this window.

Alternatively, use the “Waterfall” button to directly toggle chart modes.





Conclusions

Performing an analysis of application behaviour before servers are moved between data centres (or to the cloud) can provide an enormous benefit to the project team.

a) A data centre (or cloud) move comes with a long list of risks and unknowns.

b) The exact behaviour of an application is rarely fully known by the team that “owns” the application.

c) This type of analysis reduces unknowns and reduces risks. More so if done early in the project timeframe.

d) Revealing the exact behaviour of applications can provide early notice of:
   - Which applications/servers can be moved with no (or little) expected performance impact.
   - Which applications/servers <u>will</u> have a performance impact – and how much?
   - Exactly why applications are predicted to have a large performance impact?
   - Workable remediation options that can be proposed and implemented.
   - Remediation options that can be implemented and tested well in advance of the physical server moves.

e) A similar traffic analysis can be used to determine which servers should be moved as groups.

If you’d like to discuss the possibility of performing this type of analysis in advance of your project (or if you’ve already moved and want to determine why you now have degraded performance), please get in touch via my contact details on the next page.

Phil Storey

Phil@NetworkDetective.com.au

www.NetworkDetective.com.au

www.linkedin.com/in/philipstorey3/

@PhilStorey24

www.youtube.com/c/NetworkDetective

ask.wireshark.org: @philst