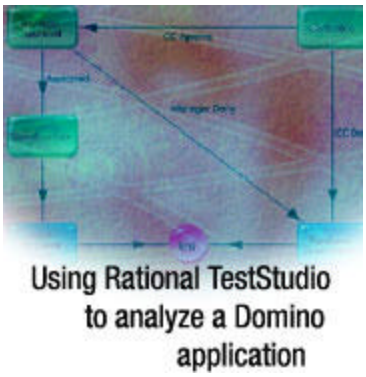


Level: Advanced
Works with: Rational, Notes/Domino
Updated: 08-Sep-2003

by Carol Zimmet
with Lewis Cote

When Rational Software joined IBM's Software Group earlier this year, there was a great sense of anticipation and expectation here at Lotus and in other areas of the organization. Rational brings a whole family of tools, methodologies, and offerings that are now available both internally within IBM and externally to our customers. Within the Lotus Engineering Test (LET) team, where our work includes solutions-type analysis, we were especially excited by these new tools and capabilities. And it hasn't taken long for us to see the benefits of using Rational tools. Although in some respects we're still in learning mode regarding all that Rational products can do, we (along with Lotus Support Services) have already used them to help address specific customer issues.

This article describes one such experience in which we employed Rational's TestStudio to help analyze and respond to the needs of a large Notes/Domino customer. This allowed us to help the customer:

- Identify a bottleneck in an application they were preparing to roll out
- Confirm their server configurations were correct and advise them on tuning adjustments
- Expedite application deployment

We start with some general background (without naming names of course) and explain how TestStudio fit into our analysis efforts with special focus on how to organize a similar load and scalability analysis effort in your environment. We found that Rational TestStudio is complete and robust and can handle many types of system analysis challenges. And you can use our experience as a guide to help analyze and respond to a similar situation at your own site.

This article assumes that you're an experienced Notes/Domino administrator, ideally with some familiarity performing system analysis with tools, such as Server.Load, Rational, or another industry-standard load and scalability tool.

Background

Our scenario involved a large international customer (based outside the United States) who planned to deploy an application. They had goals in mind for what they wanted to achieve in terms of response time and user scalability, but wanted to conduct some pre-deployment analysis of their configuration. Recognizing this analysis beforehand would be more timely and cost effective than troubleshooting and dealing with problems after deployment was underway. However, they had no simulation tools or strategy in place for simulating their application under usage. And they wanted to evaluate the performance of this application on two different platforms.

To assist the customer, we pulled together an international team with representatives from Lotus and Rational.

Members of this team included:

- A project manager responsible for identifying the customer's analysis requirements and for forming and coordinating the team to meet these requirements
- Two support representatives responsible for running the simulated user scripts and for capturing and analyzing the information reported
- Two on-site contacts to provide support, to design customer usage scenarios, and to execute tools
- A Rational expert responsible for executing Rational tools (with the workload we created) on the customer's server and more important, for analyzing load execution results

The challenges our team needed to overcome included different time zones, different skill sets, and an aggressive delivery schedule. Bear these in mind when you form your own team to attack similar issues. (Our team dealt with these obstacles and eventually succeeded in meeting our customer's goals, so it can be done.) Also note that the members had other responsibilities outside the team—a performance analysis project does *not* generally require six dedicated experts to complete!

One of this team's first decisions was to use Rational TestStudio to perform load and scalability evaluations. This represented a significant change in our previous approach toward assessing load and scalability issues. Until recently, Server.Load has been our tool of choice for creating a load against a Domino server. Rational TestStudio now provides another option. This tool can also work against the Domino server for capturing network-level exchanges and playing them back. Traffic is captured during the recording session; and depending on which protocol support you have implemented (HTTP, SQL, or Oracle), this specific traffic is filtered out. For natively supported protocols, TestStudio then generates a script from this filtered traffic. (Server.Load is generally used for NRPC, but over HTTP requests, Rational TestStudio may be an additional option. In this example, the customer's application was HTTP-based.)

Note that the choice of whether to use Server.Load or TestStudio depends on the situation; each has its advantages. In this particular case, we felt TestStudio better suited our requirements because it provided:

- Capture and playback capability
- Application simulation
- Greater flexibility in building an analysis scenario around the customer's specific application

After we selected which tool to use and which methodology to follow, our next challenge was to build a workload that performed as closely as possible to the customer's actual environment. The next sections describe how we did this.

Defining the workload and test requirements

Our customer wanted to simulate 70 to 100 concurrent users running the application. We decided to do this with a straightforward load run (with additional runs planned) using different adjustments in the environment to create different scenarios for additional analysis points. To match as closely as possible the actual user experience, we needed to capture information through a workload that simulated a user interacting with the application through a graphical user interface (a key point to consider when performing almost any load and scalability analysis that attempts to duplicate response time from an end user perspective). This last workload type is often referred to as a "probe" workload.

After reviewing the workload simulation and overall test requirements, we concluded Rational TestStudio allowed us to meet all of them. With TestStudio, we could analyze the kinds of tasks performed by users in their application usage model. These tasks included search, document manipulation, and navigation through the Web interface. We also isolated different populations of users and linked these populations to the tasks that they usually perform and to the frequency with which they did them. After the different population of tasks and frequency of tasks were determined, we created unique scripts for each user profile that point to different application components.

As mentioned previously, our customer is based outside of the U.S. Support for evaluation of international environments is key for Lotus product development efforts. Rational TestStudio tools support internationally-based application and system analysis. For example, these tools support international languages. When your default font is set to the correct language, the text for that language appears correctly. Also, multibyte data can be used in the data pool, enabling the full range of international language evaluations.

(This feature is an important requirement for any product support tool.)

Developing the workload

The methods we used to create load tests differed from what we've done in the past when we traditionally worked with a predefined scripting language. For example, we began by simulating a single user (known as a "Virtual User" in Rational terminology) and used TestStudio's record function to record a few simulated user activities. The ability to record, capture, and analyze on-the-fly is a very powerful tool and allowed us to perform ad hoc analysis and to quickly jumpstart our script development.

A key prerequisite to script development is understanding what the script needs to validate. This lets you (and/or your customer) feel comfortable with the results as you increase the number of simulated users.

The first challenge we faced in moving from a single simulated user to multiple users was the fact that we were dealing with dynamic data. For instance, unique login information was being returned from the Web interface and utilized in future application interactions for that user. We also needed to handle specific documents, which meant having to keep track of unique document information. (We'll explore the issue of handling dynamic data further in an upcoming article.)

Due to the limited time available for server access and workload development, we took a shortcut in the workload development process. Because we needed to simulate only 70 to 100 users, rather than develop scripts to handle dynamic content, we took a more "static" approach. We created unique scripts for each end user in which each user executed his own unique workflow process using the application under test. (We recognized this wasn't necessarily the most effective way to develop a workload, but we felt it was the quickest under the circumstances.) This insured that each user had a unique login and access information through HTML addresses. TestStudio supports the ability to execute multiple scripts simultaneously, so we could simulate up to 100 concurrent users.

To do this, we used Rational TestStudio to capture a script of the actions that we wanted to benchmark. We then played this script back multiple times to simulate load. We initially assumed that we could capture what we wanted to simulate via the record function, then just play it back for 100 unique users. But because the information was dynamic and specific to each test user, we needed to capture particular steps the customer wanted tested. We compiled a list of actions for which our customer requested response times, and then recorded them in a script. We then edited the script and created 100 simulated users from it.

Another dimension was added to this load simulation process when we developed a GUI application script to measure the end user response time. This second script was run on a separate workstation to capture real-world data, so the customer would know exactly what to expect when a user logged on with 100 currently active users. We could have integrated this second script into the core load script, but due to time pressures, we chose to execute it from a separate system. We enhanced this script with timing information to capture end user response time data. We placed timing parameters in various steps within the script to understand both total response time to complete the user workload scenario and also elapsed timing for individual key steps. Our GUI script is as follows:

```
Sub Main
   Dim Result As Integer

      'Initially Recorded: 5/5/2003 11:05:42 AM
      'Script Name: guidoc1
      StartBrowser "http://pieaixtest.coc.lotus.com/NewLib.nsf", "WindowTag=WEBBrowser"

      Window SetContext, "Caption=Enter Network Password", ""
      InputKeys  "abcde{TAB}password"
      PushButton Click, "Text=OK"

      Window SetContext, "WindowTag=WEBBrowser", ""
      Window WMaximize, "", ""
      StartTimer "AppStep1"
```



```
    Browser SetFrame,"Type=HTMLFrame;HTMLId=Tree",""
    Browser NewPage,"","",""
    HTMLImage Click, "Type=HTMLImage;Name=FR1", "Coords=8,8"
    HTMLImage Click, "Type=HTMLImage;Name=FC1", "Coords=7,9"
    Browser NewPage,"","",""
    HTMLImage Click, "Type=HTMLImage;Name=FC1C1T1", "Coords=7,11"
    HTMLImage Click, "Type=HTMLImage;Name=FC1C1T2C2T0", "Coords=7,10"
    HTMLImage Click, "Type=HTMLImage;Name=FC1C1T2C2T1C3T0", "Coords=8,10"
    StopTimer "AppStep1"
    StartTimer "AppStep2"
    HTMLImage Click, "Type=HTMLImage;Index=50", "Coords=12,8"
    StopTimer "AppStep2"
    StartTimer "AppStep3"
    Browser SetFrame,"Type=HTMLFrame;HTMLId=Bottom",""
    Browser NewPage,"","",""
    HTMLImage Click, "Type=HTMLImage;Index=31", "Coords=3,6"
    StopTimer "AppStep3"
    'close app
    Window CloseWin, "", ""

End Sub
```

One final challenge was the fact that the customer's native language was not English. As a result, when creating the scripts, we had to connect to a Web site whose language base was non-English.

Executing the workload

One thing we had to consider when executing the workloads was the size and type of client used to run them. In this case, we used a single client driver to simulate the 100 users. This client driver was an IBM A31p laptop (2.4 GHz PC, 1 GB RAM, 30 GB of free disk space), running Windows 2000 Service Pack 3 with Internet Explorer 6 Service Pack 1. (Note that this computer was not maximized for performance.)

Note that Rational requires minimum configurations to run their tools. Therefore, the design and implementation of the simulated workload has an impact on the number of users that can be simulated on a single client. As a rule of thumb, you can usually simulate several hundred users from a single client driver, depending on the workload and system configuration (memory being the key).

At this point, we had a series of scripts logging into the application with unique user IDs and navigational flow to simulate the required user scenarios. The GUI script simulated a single user clicking buttons in a browser and timed how long it would take to complete an action. An example of the workload display screen appears as follows:

The workloads were both Virtual User-based and a single GUI-based script as described in the previous section.

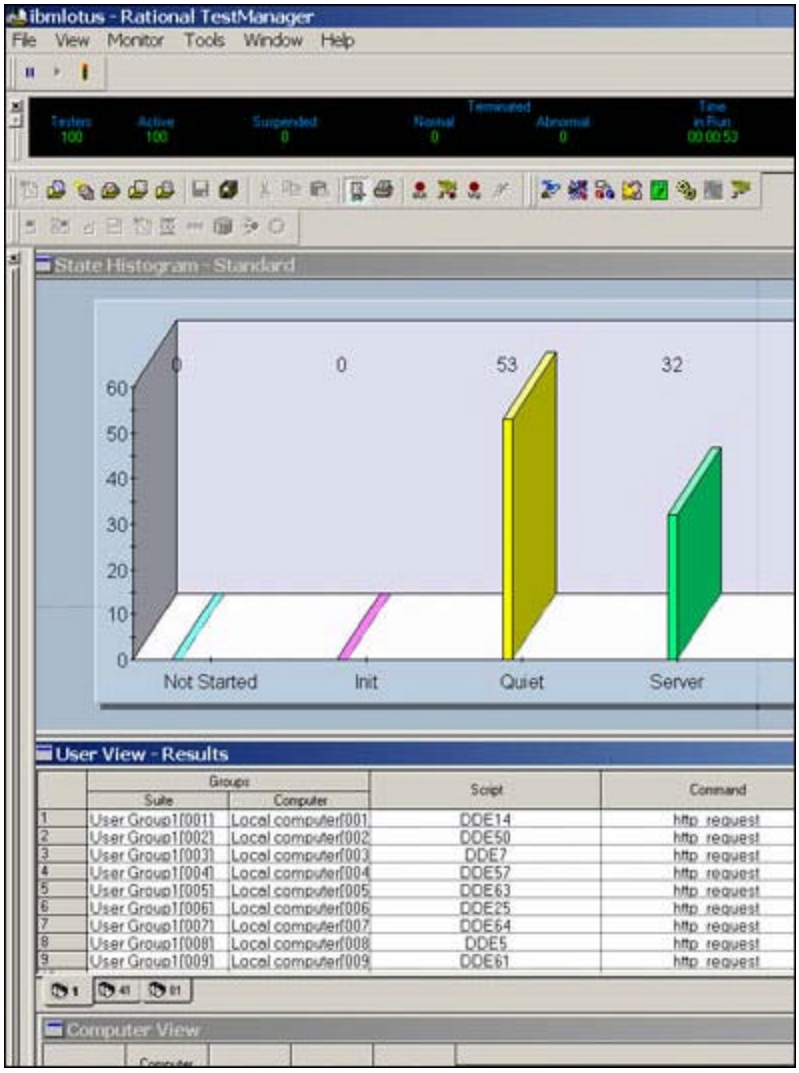
Our next step was to gradually increase the number of simulated users until we observed an increase in the response time that indicated that the additional simulated users hit a boundary condition or bottleneck that impeded optimal user response time.

Analyzing the results

We eventually observed time delays in user response by allowing the workload to run over a period of time and/or by incrementally adding users. The time delay information was captured via the GUI-based script, representing user response impact. This was the point at which application and platform metrics were captured to identify where the slowdown occurred. After we analyzed the data, we made adjustments at the application and platform levels to help tune and optimize. We performed this step multiple times (and in theory could have continued to repeat it indefinitely, tweaking settings and measuring the results).

In the real world, it is best at this point to have established a goal of minimum acceptance to know what you are working towards. From a general strategy, this can be an improvement in the number of users, an increase in the number of transactions executed, or another factor. In our case, we executed the workloads, incrementing the number of users. When we observed user performance degradation, the customer's engineers reviewed the application and system information we captured and made the appropriate adjustments. This was a significant effort. There were many different parameters that could be adjusted on the application and many different types of platform settings that could also be adjusted. Because we took the approach of adjusting only one setting at a time (or a small number of them) and then measuring the impact, you can appreciate the level of complexity for making the correct assessment and the importance of determining which changes should be made to have a positive impact.

While a test script is executing, you can monitor its progress. The following screen shows the interface that displays the script's progress:



This screen displays (among other data):

- The status of the test in execution
- How the test is progressing, monitoring commands executed, and the state of simulated (virtual) users
- Opportunities to make adjustments

The graph helps confirm your user profile activities—a good summary to ensure your simulated users are executing in the expected state. In this example, users are operating in the quiet server or code states.

Delivering the final results

As typically occurs at the end of a load and scalability analysis, we generated a detailed report. This report included data about:

- Workload executed
- System under test, including hardware specifications and software installed
- Results observed, including platform data
- Configuration file recommended
- Tuning steps executed

You can see an example written test report containing similar information by visiting the NotesBench Consortium Web site. You can use this report as a model for your own analysis reports. The NotesBench site contains examples of many reports generated by platform vendors on behalf of their simulated Domino workloads. (These reports use the NotesBench scalability tool, but Rational TestStudio reports contain similar information.)

Conclusion

This article reviewed the questions and issues you typically need to consider when conducting a performance analysis. and it discussed tools and methods for addressing these issues. In addition, we discussed the different skills you should have on hand for an analysis of this type. You can use our experience to help guide your own analysis projects.

Our experience confirmed that:

- Rational tools are flexible and feature rich. They provide a variety of strategies you can use to undertake an analysis to understand load and scalability issues around customer applications and environments.
- At the same time, our recommendations concerning load and scalability analysis remain the same, no matter which tool is used: You need to have a baseline of output so that you have something to compare to as your environment changes.

Demo versions of Rational tools are available from the Rational product page. Take them out for a test drive and see how they can work for you!

ABOUT THE AUTHOR

Lewis Cote is a Systems Engineer at IBM Rational. Lewis started out at a company called SQA as a Technical Support Engineer supporting automated testing tools. SQA merged with Rational Software, and at the time, Lewis held the position of Technical Support Team Leader, then moved onto a Technical Support Manager for the performance testing tools. Currently, Lewis is a Technical Representative for Brand Services located in Lexington MA, focused on automating testing tools.

ACKNOWLEDGEMENTS

The following individuals played key roles in the customer analysis project described in this article:

Bruce Walenius, EMEA Critical Situations Manager, spearheaded this project. To identify and address the customer's needs, Bruce pulled together an international cross-family team with representatives from different product-related areas in Lotus and Rational. This was an impressive feat as our relationship with Rational was newly forming. Bruce helped forge a new connection channel between the teams.

Ivan Dell'Era, then part of the Lotus Support Team, worked with Lewis Cote in running the simulated user scripts and in capturing and analyzing the information reported.

Franck Horowitz, IBM IGS Europe, executed the Rational tools with the workload on the customer server.

