Performance Testing Your SAP System — Strategies for Ensuring a Successful Test Execution

Tim K. Nguyen

Tim K. Nguyen is Managing Principal of Orchid Consulting Group, Inc., a consultancy that focuses on SAP and e-Business systems performance and business continuance issues and solutions. With over 20 years of experience building successful IT professional services practices, Tim oversees most of Orchid Consulting's projects worldwide.

(complete bio appears on page 72)

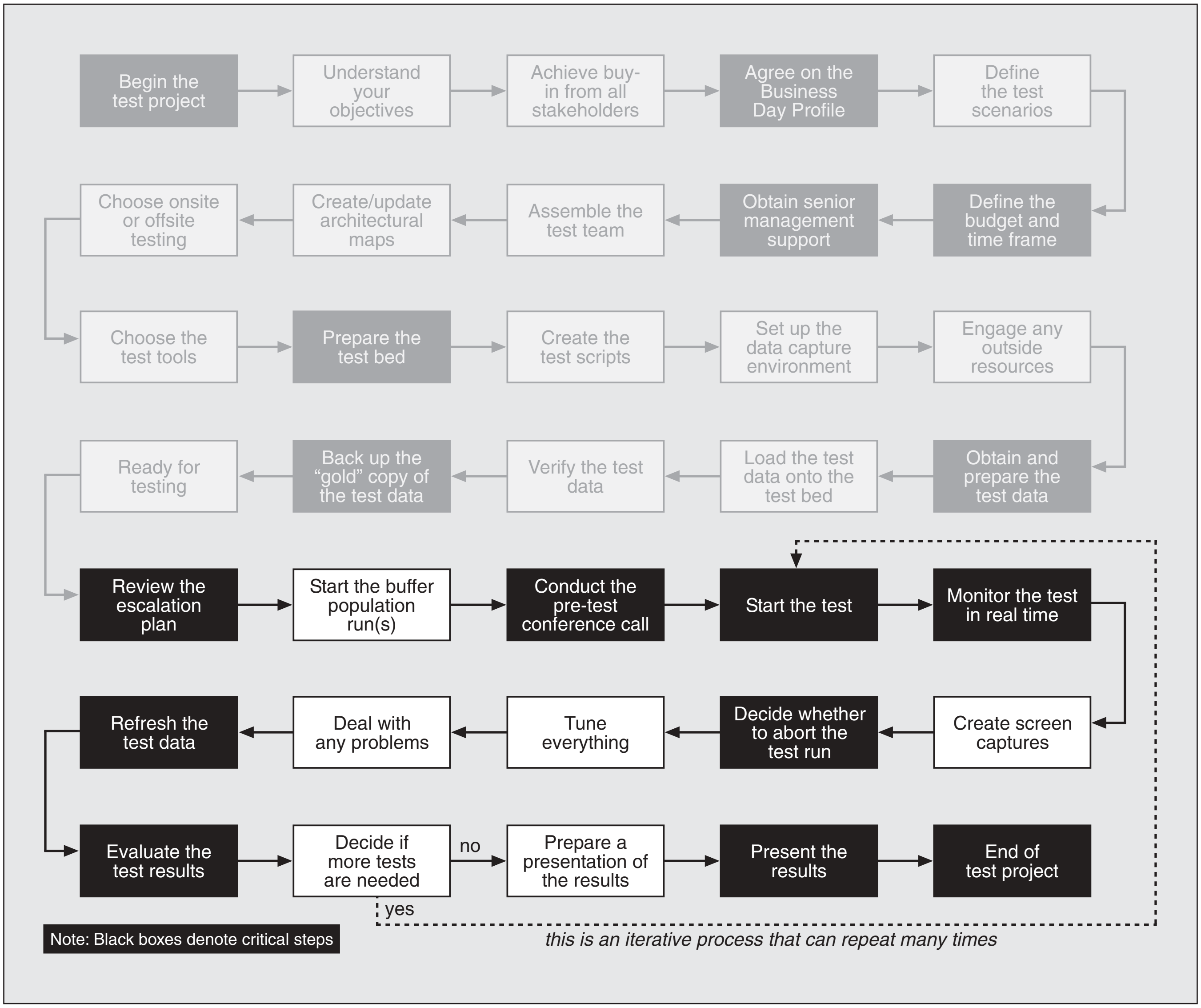
"Learn by doing" is a time-honored maxim, but when it comes to performance-testing your SAP system, there are some things you'd rather not learn by doing wrong. A performance test project involves many highly skilled people from both inside and outside your organization. Once test execution begins, you can't afford to have these people standing around idle or not using their time efficiently. While a certain amount of downtime is unavoidable for both people and systems, it *can* be minimized. Successfully executing efficient performance tests requires discipline, leadership, quick thinking, and even a bit of luck. Most of all, it requires thorough planning and the right testing strategies.

This is the second article of a two-part series on performance testing SAP systems. The first article[1] covers the preparation phase — what you're testing and why, putting together the right testing team, defining test scenarios, selecting testing and measurement tools, and properly setting up the test environment. This second installment picks up where the first leaves off.

In these pages, you will learn how to develop your escalation plan, strategies for minimizing downtime, how to apply your testing tools, how to present your test results to management, and more. And every recommendation, technique, and strategy is real-world-tested, so you can reap the benefits of my experience leading testing teams and avoid some of the harder lessons that come with learning by doing.

[1] "Performance Testing Your SAP System — Best Practices for Preparing Your Test Environment" (January/February 2004).

Figure 1 *Performance Test Project Tasks*



The Execution Phase of Testing — Overview

The execution phase of testing consists of many tasks. The diagram in **Figure 1** shows them in the context of the overall project (the preparation steps covered in my previous article are "grayed out"). Despite the linear presentation in the chart, these tasks don't necessarily take place serially — some tasks overlap or take place simultaneously, and many tasks are repeated multiple times. Each test run creates its own demands, so the tasks performed for a run may vary from test to test. For the purposes of this discussion, however, I will categorize the execution phase tasks as follows:

1. Create a skeleton for the escalation plan (to be fleshed out in the pre-test meeting).

2. Perform pre-test activities and conduct a meeting before each test run.

3. Populate the buffers (if necessary).

4. Apply testing and measurement tools.
5. Observe tests while they run.
6. Tune the system.
7. Deal with problems and contingencies.
8. Refresh data between tests.
9. Change the test environment between tests.
10. Evaluate the results.
11. Prepare data for management review.

Over the course of the rest of this article, we will take a detailed look at each of these steps in the order shown.

Task #1: Create a Skeleton for the Escalation Plan

As part of the preparation for the test project, either the overall project manager or the test run manager should create a "skeleton" for the project escalation plan. The escalation plan outlines not only what to do and who to call when and if things go wrong, but it also includes some type of checklist and even a list of known error conditions to help those monitoring the test determine just how wrong things have gone. A clearly defined escalation plan is essential for ensuring calm, appropriate responses to problems during testing. It helps keep small problems from turning into big ones, and minimizes damage if large-scale problems do arise.

The skeleton escalation plan should be in the form of an outline that lists possible issues (these will be the top-level items of the outline). Beneath each issue should be a series of questions to be answered by members of the testing team during the numerous pre-test conference calls or kickoff meetings — questions such as "What should we do if a process under *SM50* appears to be running for too long?" or "Who should we call and what should we do if *SM21* and *ST22* show ABAP dumps during the test run?" and "Who makes tuning changes on the disk storage system?" (Note that this is by no means an all-inclusive list of the questions that should be asked.) To flesh out the plan, as it is reviewed, team members should step forward and offer their services for solving particular problems.

You may be wondering why I don't recommend simply assigning responsibilities. In my experience, those best suited for resolving certain problems rarely report directly to the person who creates the skeleton escalation plan, so assigning responsibilities is problematic. If your test preparation is thorough, however, all team members will clearly understand the objectives of the test project and its importance, and will be willing to assume responsibility for resolving problems in their areas of expertise. If, as the project manager, you do not see the obvious candidates stepping up, then you will need to find ways to enlist their support. For example, you can do something as simple as explaining the test objectives again, something as extreme as having a conversation with their managers, or something in between.

Typically, you structure the escalation plan in ascending order of problem severity, with minor problems listed first, followed by mid-level problems, and finally major problems:

- Minor problems include things like incorrect test loads or minor network issues — problems that are likely to occur, but can normally be fairly easily resolved without calling in high-level experts.

- Examples of mid-level problems include redo logs that are filling up the SAPARCH directory too quickly or "database full" problems; these are classified as mid-level because they can be resolved quickly to allow the test to continue, but require the involvement of senior-level experts.

- In my view, problems due to incorrect parameters in SAP instance and database profiles, as well as problems associated with data corruption, should be classified as severe, since they will make continuing a test run impossible.

For each issue listed, describe the situation that might occur and then clearly show a plan of action — in other words, the plan should indicate who is responsible for resolving the problem, and the specific steps to take. Then list what to do and who to contact if this attempt to resolve the problem is unsuccessful, guiding the "first responders" up the chain of command to the appropriate level of senior expertise.

> ✓ ***Tip***
>
> *I cannot recommend a single format or tool for creating a good escalation plan. Escalation plans can feature flow charts, or can be Microsoft Excel spreadsheets or Microsoft Word tables. I generally use the table and outline functions of Word to create the skeleton for an escalation plan. With Word you can also easily save documents as HTML files for posting on the project's intranet.*

Things to Plan For

While an all-inclusive list of possible error conditions is impossible to compile due to the range of possible tests and test environments that exist, here are some situations I have encountered that stopped in-progress tests until the cause was resolved (responses to these problems are outlined in the upcoming section "Task #7: Deal with Problems and Contingencies"):

- Too much swap on both application and database servers
- Too much load on a particular application server
- Incorrect test load being applied (this is a variation of problem 2 — instead of one server being overloaded, you have a group of servers that are very busy while others are idle)
- Very long-running transactions under transaction code *SM50*
- Problems with the database server
- Problems with external links
- Data-capture problems
- Network problems
- Failing batch jobs

In addition to problems that may occur during a test run, there are a myriad of pre-test problems, such as the SAP system or a database not coming up after you have made changes to instance or database profiles. All problems, whether they occur during pre-testing or testing, should be included in the escalation plan. We will discuss some of these problems later on, along with recommendations on how to overcome them.

Task #2: Perform Pre-Test Activities and Conduct a Meeting Before Each Test Run

As with the planning phases of testing, clear communication among team members before each test run is vital, especially in the days and hours leading up to the first test run. Considering the money and resources allocated to the project, you don't want to rush into testing without being absolutely certain that everyone is prepared and clearly understands both the test plan and their own parts in it.

Schedule a mandatory meeting for all test participants at which the test plan will be completely reviewed, including the roles and responsibilities of each team member. If you have a small team based at one site, you can hold a working lunch or some other kind of in-person meeting to go over last-minute plans and responsibilities. If you work for a multinational company, it's likely your test team is geographically dispersed; in this case, you need to schedule a conference call. In fact, I have found that even if everyone is in the same building, a conference call is still the best way to meet, because everyone can participate from their own desks, where they have easy access to files, documents, calendars, and anything else they may need to refer to during the meeting.

In my experience, the following items need to be covered during pre-test meetings (plan on covering all of them thoroughly in the initial pre-test meeting; subsequent meetings may only require an abridged form):

1. **Set expectations.** The test execution phase can be stressful, so it helps if everyone on the team can enter it with similar expectations. Your entire team must understand that, typically, only about 40% of their time will be spent conducting the actual tests; another 40% will be devoted to problem-solving (despite all your preparations!), and the remaining 20% will be consumed with the data refresh activities necessary to conduct subsequent test runs.

2. **Review roles and responsibilities.** Go over the list of roles and responsibilities created in the test planning phase (this was discussed in detail in my previous article). Confirm that each participant is clear on his or her responsibilities during testing and has completed all preparations. Remind the team that the testing itself is not the biggest part of the project (see item 1), which is why they need to be ready to respond to any problems (you can't stress this point too often).

✓ *Experience Teaches…*

I once was forced to delay test runs because a database administrator did not fully understand his crucial role, which was to change the Oracle database parameters on the second series of tests, when a lot more memory was to be added to the database server. That DBA mistakenly thought he was needed for emergencies only and did not allocate time for making changes to the database parameters, nor for actively participating in the test under the changed environment to ensure that there were no problems. Misunderstandings like this lead to bad starts and can have a negative effect on the enthusiasm and morale of the team. The lesson is that it's critically important to review all roles and responsibilities as part of the pre-test meeting so that everyone is on the same page.

3. **Review the test plan in detail.** If you followed all the steps laid out in my previous article, you have a well-developed, logical, and complete test plan, and your team is already quite familiar with it. Nevertheless, if there are any gaps, problems, or questions regarding the test plan, address them now. If you plan to have multiple test scenarios, it is a good idea to review *all* the scenarios with the team and answer any questions team members might have.

4. **Review strategies for handling problems.** Emphasize that rapidly detecting problems is the first step toward rapidly resolving them. Go over the following problem-handling rules:

   - **Pay attention to anomalies and alerts.** During the test, if something doesn't look right — whether it's data, performance indicators, or other metrics — it probably isn't. An experienced testing team will have a good feel for values that are reasonable, and if they detect an anomaly, something is probably wrong with the test. If the testing tools create an alert message, you definitely have a problem. Detecting problems requires focusing on what is happening during the test, so emphasize that all tests must be monitored by attentive team members, not left to run unattended. The test manager must fully concentrate on monitoring the test and interacting with other members of the monitoring team. A part-time, unfocused test manager is a sure recipe for a failed test.

   - **Stop an unsuccessful test immediately.** Depending on the size of the database and other system factors, test runs can take hours. Preparing the data, backing it up, restarting the system, and all the other aspects of the test add even more time to its duration. Emphasize to your team members that if something is wrong with a test run, they should advise the test manager so that he or she can stop it immediately. Letting the test run to completion only wastes time if the results are unusable, and it could put you days behind schedule.

- **Document all problems and issues.** You'll have to remind team members about doing this again and again, because it's so easy to just make a minor correction to a test or tweak system settings to fix a problem, and then forget to document what you did. Documentation of problems (whether on paper or electronically) is essential for finding clues to bigger system problems, knowing exactly what was tested and how, and conducting successful tests in the future. Emphasize to every member of your team that screenshots must be taken of every problem, and that everything they do must be written down clearly and completely and incorporated into the final testing reports.

5. **Flesh out/review your escalation plan.** The escalation plan is at least as important as the test plan. Remember that not only does your escalation plan have to be carefully developed, but also that all potential responders need to know and agree with their roles in resolving problems.

Task #3: Populate the Buffers

It is well known that a newly rebooted SAP system will have poor response time because data has not yet been read from the database into the memory of the SAP application servers, a process that techies jokingly call "cooking the buffers." In fact, some SAP systems may need as much as four to six hours of heavy usage to reach an optimum performance level.

Therefore, anytime you have modified SAP instance parameters, made changes to system hardware, or restarted the system after a shutdown, you will need to execute at least one or two "throwaway" test runs to fill the buffers for all servers involved in the test. If you don't, your response time may be unrealistically low. Be sure to factor in time for this activity in your test planning — you'll need to fill the buffers every time an SAP application server is rebooted, which can be quite often, especially if you are trying to test various new hardware configurations.

> ✓ ***Note!***
>
> *If your actual test routine will make changes to the database, you will need to design some special throwaway test runs that do not change the database, in order to properly cook the buffers of the database server.*

Whether or not the buffers are cooked just right depends on many factors. One way to tell is to compare the response times in the most recent throwaway test run to the response times in the previous run — when there is only a small difference between them, which indicates that the systems are no longer fetching the same data from the database, the buffers are probably filled.

Task #4: Apply Testing and Measurement Tools

In this section, I discuss how to use certain Computing Center Management System (CCMS) transactions to capture test data,[2] and I also briefly touch on the Microsoft Windows PerfMon and VERITAS Luminate testing tools.

> ✓ ***Note!***
>
> *Unless you are familiar with setting up test counters and know what you are looking for, you will need outside experts to assist you with the setup of the data-capture environment and the real-time monitoring and capture of the data. Once the test runs are complete, these outside experts can also help you interpret the results.*

[2] Evaluating and choosing your testing tools is part of your test preparation. This process was discussed in detail in my previous article.

Using CCMS Transactions ST03 and ST03N to Capture Test Data

ST03 and *ST03N* are well-known workload analysis transaction codes, the stock-in-trade of SAP system performance tuners, who generally prefer one or the other. My preferred tool is *ST03*.

My favorite technique for checking the load being put through an SAP application server involves the *Last minute load* button:

1. In transaction *ST03*, click on the *Last minute load* button, and choose to analyze 15 minutes of workload, which takes you to the main *ST03* screen.

2. Using the pulldown menu, select *Goto* → *Summary reports* → *Workload overview*, where you will get a vast amount of performance information on time profiles, response time distribution, transaction/report profiles, top dialog steps by response time, top dialog steps by database requests, and much more. The best part is that you can download *all* this data into a file for later review and documentation (see the next step).

3. To download the data, select *Workload* → *Download*, choose a file format, and specify a file name. I strongly recommend downloading the data into HTML format, although other formats, such as unconverted, spreadsheet (XLS), and RTF, are also supported.

Remember, since this technique retrieves “last minute load” information, you *must* perform these steps in real time, while the test is running. It helps to have several team members available for this work if there are lots of application servers involved. If you are unable to capture all of the *ST03* data in real time, you may be able to capture it using the *Previous days* button, where you can view the captured data stored in the SAP database according to a specified time window. The steps are pretty much the same as those for using the *Last minute load* button.

Using Microsoft Windows PerfMon to Capture Test Data

The Microsoft Windows PerfMon tool offers almost two dozen data points (also referred to as “counters”) that can yield information on CPU load level, swap activities, network traffic, memory usage, I/O wait and I/O busy status, and much more. Exactly which counter to set and at what threshold depends on the objective of the test run, and this is where you may require outside expertise in Windows test instrumentation.

Figure 2 is a table of some counters I have found useful during performance tests. An example

Figure 2 *Selected PerfMon Counters Useful in SAP Performance Tests*

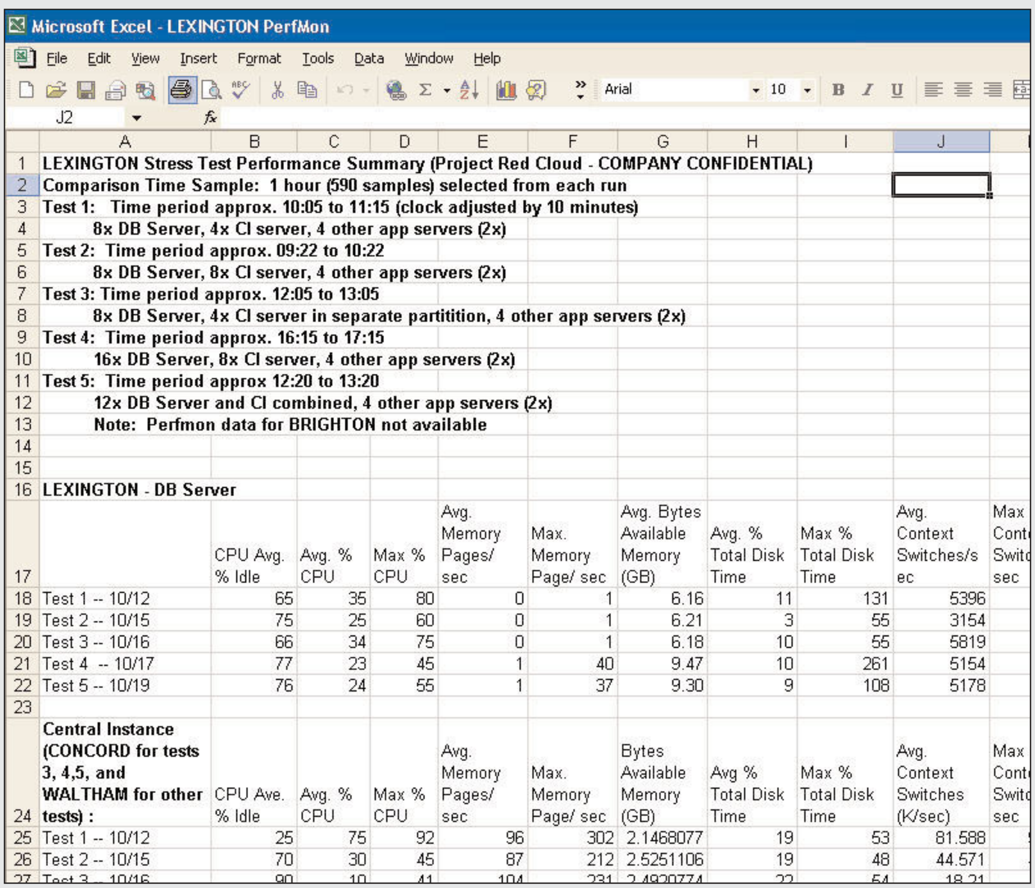
Counter Name	What It Does	How It Can Be Useful
CPU Avg. % Idle	Measures the average percentage of time that the CPU is waiting for work.	Tells how much headroom is left on the server for taking on more workload.
Avg. % CPU	Measures the average percentage of time that the CPU is working.	Tells how busy the CPU has been throughout the test run, and therefore whether or not the server is close to being maxed out.
Max % CPU	Indicates the spike in CPU usage in a particular server.	This important counter indicates how a particular test load is affecting the CPU. Even if Avg. % CPU is low, a high Max % CPU (over 95%) is a cause for concern, and should be investigated.

(continued on next page)

Figure 2 (continued)

Counter Name	What It Does	How It Can Be Useful
Avg. Bytes Available Memory (GB)	Indicates the average amount of memory on the server available to run applications.	Tells how much memory is available on a particular server. Situations differ, but in all cases a low number is not good.
Max % Total Disk Time	Indicates how many I/O activities have occurred during a test run.	It depends on the situation, but a high number could indicate anything from datapaths that are too busy to excessive disk swapping.
NICs Avg. Total Bytes/sec	Indicates how much data has been sent on average over a particular network card.	This is an important counter that provides information on network traffic and is very useful in tuning for proper network load balancing.
NICs Max Total Bytes/sec	Indicates how much total data has been sent over a particular network card.	This is another important counter used to identify performance bottlenecks.
MemPg/sec	Indicates how much paging and swapping occurred on a server.	Excessive paging and swapping is a sure sign of a potential or actual performance problem.

Figure 3 *An Example Summary Report of Compiled PerfMon Data*



	A	B	C	D	E	F	G	H	I	J	K
1	**LEXINGTON Stress Test Performance Summary (Project Red Cloud - COMPANY CONFIDENTIAL)**										
2	**Comparison Time Sample: 1 hour (590 samples) selected from each run**										
3	**Test 1: Time period approx. 10:05 to 11:15 (clock adjusted by 10 minutes)**										
4	**8x DB Server, 4x CI server, 4 other app servers (2x)**										
5	**Test 2: Time period approx. 09:22 to 10:22**										
6	**8x DB Server, 8x CI server, 4 other app servers (2x)**										
7	**Test 3: Time period approx. 12:05 to 13:05**										
8	**8x DB Server, 4x CI server in separate partitition, 4 other app servers (2x)**										
9	**Test 4: Time period approx. 16:15 to 17:15**										
10	**16x DB Server, 8x CI server, 4 other app servers (2x)**										
11	**Test 5: Time period approx 12:20 to 13:20**										
12	**12x DB Server and CI combined, 4 other app servers (2x)**										
13	**Note: Perfmon data for BRIGHTON not available**										
14											
15											
16	**LEXINGTON - DB Server**										
17		CPU Avg. % Idle	Avg. % CPU	Max % CPU	Avg. Memory Pages/ sec	Max. Memory Page/ sec	Avg. Bytes Available Memory (GB)	Avg. % Total Disk Time	Max % Total Disk Time	Avg. Context Switches/s ec	Max Cont Swit sec
18	Test 1 -- 10/12	65	35	80	0	1	6.16	11	131	5396	
19	Test 2 -- 10/15	75	25	60	0	1	6.21	3	55	3154	
20	Test 3 -- 10/16	66	34	75	0	1	6.18	10	55	5819	
21	Test 4 -- 10/17	77	23	45	1	40	9.47	10	261	5154	
22	Test 5 -- 10/19	76	24	55	1	37	9.30	9	108	5178	
23											
24	**Central Instance (CONCORD for tests 3, 4,5, and WALTHAM for other tests) :**	CPU Ave. % Idle	Avg. % CPU	Max % CPU	Avg. Memory Pages/ sec	Max. Memory Page/ sec	Bytes Available Memory (GB)	Avg % Total Disk Time	Max % Total Disk Time	Avg. Context Switches (K/sec)	Max Cont Swit sec
25	Test 1 -- 10/12	25	75	92	96	302	2.1468077	19	53	81.588	
26	Test 2 -- 10/15	70	30	45	87	212	2.5251106	19	48	44.571	
27	Test 3 -- 10/16	90	10	41	104	231	2.4920774	22	54	18.21	

summary report of compiled PerfMon data is shown in **Figure 3**.

Using VERITAS Luminate to Capture Test Data

VERITAS Luminate is a tool that captures performance data on SAP systems and stores everything in a data warehouse for later analysis and review. Please refer to my previous article for a detailed discussion of this tool and how it can help you with your testing efforts.

Task #5: Observe Tests While They Run

You have an escalation plan, and you've impressed upon team members the vital importance of stopping a test run when there is a problem and implementing the escalation plan as quickly as possible. If the test isn't run in real time and monitored while it's running, however, no one will know to stop it when a problem comes up. For example, there might be an Oracle trigger problem causing ABAP dumps — you need someone regularly scanning transactions *SM21* and *ST22* to catch a problem like this.

Observing test runs is the best way to minimize the effects of problems. It's also the most effective way to produce test documentation. For example, I've learned that creating the screenshots required to document the various parameters or values achieved during each test is a lot easier to do in real time as the test is being executed rather than after the test is done. Even though you think you'll remember everything, by the time you start to write up the documentation, certain details may get hazy. Be sure to annotate the screenshots you capture so that you can accurately report all the necessary details of the test when it is complete.

Another reason to monitor test runs in real time is that certain transactions — like *ST03* and *SM50* — can yield a treasure trove of valuable information when you use them to capture data in real time. The same can be said about tools like Microsoft Windows PerfMon and VERITAS Luminate. For example, looking at PerfMon graphs in real time can point you to excessive swaps on the database server or network traffic that is overwhelming a particular application server.

✔ ***Experience Teaches…***

Think about <u>who</u> will be observing the test runs. On more than one occasion, I have seen power users observing a test react with surprise and amazement as they get a complete picture of what happens to the system when the Business Day Profile[3] workload is applied to it. I also know that more than one SAP functional configuration enhancement has resulted from lessons learned during test runs. I suppose you could say that such configuration enhancements are the "side benefits" of test runs.

Task #6: Tune the System

The tuning of hardware and network settings, instance profiles, database parameters, and various other settings takes place before, during, and (frequently) after testing occurs. Prior to running tests, collect baseline values on all of these settings to make sure they are within established specifications. Depending on the type of testing you are doing — evaluating the system's response to increased volume, for example — some of these settings may require adjustment or attention during test runs.

[3] The Business Day Profile consists of SAP business transactions that are run in a typical business day. For more details on defining your Business Day Profile, please refer to my previous article.

Because your tuning requirements depend entirely on your unique system and what you are testing, a comprehensive list of settings to tune would not be very helpful. Here, however, are a few examples of settings that had to be tuned during test projects I was involved in:

1. **Various SAP instance profile parameters:** These parameters determine the "personality" of an SAP application server. Changing one or more of them can drastically affect performance. Tuning SAP instance profile parameters is highly specialized work that often requires the involvement of the SAP EarlyWatch group.

✔ *Experience Teaches…*

I have been in situations where adding more memory and CPUs to SAP application servers did not significantly change performance, but making selected changes to the SAP profile parameters (in one case, changing the sizes and behavior of the ABAP heap) yielded stunning improvements. Again, tuning the SAP profile parameters is a highly complex exercise, and in many cases will require the involvement of SAP Remote Consulting or EarlyWatch experts.

2. **Various database parameters:** There is no question that if you add more horsepower to your database server (more memory, more CPUs, etc.), you will need to tune database parameters — memory allocation, size of the shared global area, size of the buffers, and so on — so that your database server will run optimally. Once again, tuning a database is a complex endeavor that will require skill and expertise.

3. **Hardware, network, and operating system parameters:** Depending on the issues uncovered during the test runs — often by looking at PerfMon results — you will most likely need to tune these parameters as well. The assistance of hardware vendor experts is not required for this endeavor, but it is very useful.

System Tune-Up Checklist

Check and monitor:

- ☑ Database parameters
- ☑ Instance profiles
- ☑ Network settings
- ☑ Hardware settings
- ☑ Other software parameters
- ☑ Operating system parameters

The most important items to agree on with your team regarding performance tuning are:

- What tuning work should be part of the test runs
- Who will do the tuning

Again, these decisions will be highly dependent on your particular system environment and what you are testing, but let's say you want to test the overall system performance under a larger database server. In this case, the SAP instance profile parameters on the SAP application servers should not be touched, and only the database administrator and hardware vendor experts should tune the database parameters and database server.

It's critical to engage the support of your technical experts *before* you need their help. As discussed in my previous article, you must anticipate whose expertise you may require for tuning the system and make arrangements to call on them for immediate response during testing if necessary. In some cases, you may need to enlist outside help. Further, the test results may indicate a need to undergo system tuning for improved performance — so lining up experts in advance will help you swiftly implement needed changes.

Remember that if any tweaking or tuning becomes necessary during the tests, your testing will be temporarily suspended until the tuning is complete. This could mean an hour or a day, depending on what's being done and the availability of the resources required to perform the corrective actions. Prepare your team for the possibility of downtime, and have a plan for putting that time to good use. Ideally, team members whose time is 100% committed to the tests can stay productive by working on documentation or setting up the next test runs.

Task #7: Deal with Problems and Contingencies

In my experience, problem solving requires as much time as actually running the tests, even on projects with very well-planned tests, which is why you must make it very clear to your team right from the start that they should expect to spend about 40% of their time solving problems. Spending a great deal of time on problem solving does *not* indicate that the tests are unsuccessful or that the preparation was inadequate.

Although problems are inevitable, you can take steps to minimize their occurrence and speed their resolution when they do crop up. Think of it as making sure your spare tire is properly inflated before a trip, checking your oil and coolant, and throwing a few tools in your trunk before setting off.

Figure 4 lists some of the most frequently encountered problems, in my experience, along with tips and techniques for preventing and resolving them.

Figure 4 *Problem-Solving Tips and Techniques*

Too much swap occurring
☑ Verify that SAP instance profile parameters, database parameters, and operating system parameters are properly configured. ☑ Ensure that there are no errors in memory configurations.
Necessary data not captured
☑ Check the network connections to data-capture workstations. ☑ Verify that counters are set properly and started as the test is launched.
SAP system does not come up after profile parameters are changed
☑ *Always* have a backup copy of the current working profiles (including global, startup, and instance profiles). With these "good" copies, you can at least restore the system to its previous state, and then begin the troubleshooting and problem diagnosis process.

(continued on next page)

Figure 4 (continued)

Test load not correctly distributed

- ☑ Review test scripts for logic and execution procedures.
- ☑ Check to see if there are network or hardware problems with one or more of the test-load-generation workstations.

Here are two examples of incorrect test loads and what you can do about them:

1. **Too much load on a particular application server:** In this situation, the CPU utilization on one application server is near 100%, while the other application servers do not exhibit the same load level. If the condition persists for 10 to 30 minutes, it is possible that the test scripts were not configured correctly to distribute loads across all servers. There is, however, an exception — if an application server is dedicated to running heavy batch programs, it is possible for that server to be quite busy while other servers are not.
2. **Incorrect test load being applied:** Instead of one server being overloaded, you have a group of servers that are very busy, while others are idle. This situation is the reason it is so crucial to have test script creators participate in the test run — the test script analyst may have decided to intentionally overload some servers to accurately simulate conditions defined in the Business Day Profile. So, instead of stopping the test run, consult with the test script creator to determine if anomalies are intentional.

Database errors

- ☑ Have the database administrator review the database to see if there are dropped tables corrupting the data and to determine whether the errors affect the test run.

For example, let's say database errors are popping up all over the place in transactions *SM21* and *ST22*, and your ABAP programs are dumping left and right. In my view, unless the database error is well known or understood, the test must be stopped at once. The causes of database errors are too many to list, but they include tablespaces being full, redo logs filling up the SAPARCH directory, and tables being destroyed. Your database administrator needs to immediately restore the "gold" copy of the database — there is no point in trying to repair the corrupted database, since it is not a production system. You need only understand why the errors occurred, and then restart the test run.

Oracle redo logs filling up

- ☑ Have the database administrator archive or clear the redo logs. In some cases, it is possible to run the tests with the Oracle redo logs turned off, but your database administrator must have the final say on the matter.

Database filling up

- ☑ Have the database administrator add more data files.

Data refresh problems

- ☑ Replace defective hardware.

Figure 4 (continued)

Network problems
☑ Involve the network team and replace defective hardware, or adjust the appropriate network settings. There are two kinds of network problems: 1. **A problem on the general end-user network:** This type of problem can affect network traffic between the test load workstations and the application servers, or prevent data from being captured by the data-capture workstations. I had one experience with the former in which the principal user network was rendered practically unusable by the SoBig virus, but there may be other possible causes of general user network problems, ranging from hardware failure to misconfigured network switches. 2. **A problem on the private network between the SAP application servers and the database server:** Any abnormal network condition will render your test results meaningless. This problem is often caused by a hardware malfunction, such as a bad port of the Ethernet switch or hub, or a faulty Ethernet card.
Batch jobs on dedicated batch server don't run
☑ Use transaction *SM37* to check why batch jobs are failing, and log on to the dedicated batch job server to diagnose problems. Note that if batch jobs are a required element of your Business Day Profile, then the test run *must* be stopped if key batch jobs have failed.
Long-running or hung transactions
☑ Check the system log with transaction *SM21* to see what problems occurred, and then take the appropriate corrective action. ☑ If you made changes to one or more parameters in the SAP instance profile, it is possible that those changes may be causing this condition, but such changes are not the only reasons for this type of problem. If the running time of a dialog or batch process is quite long, and that process's flag is red in transaction *SM50*, then there is a good chance that the process has hung in some way. You must stop the test run and find out why the process hung or took so long to complete.
Problems with external links
☑ Check with the network team, and use transaction *SM59* to test the RFC connection. If your test requires a connection to an external system, such as TIBCO, webMethods, MRP, or the like, and the link fails, your test results may be invalid.

Task #8: Refresh Data Between Tests

Some of your tests will require a data refresh either before the test or between tests. In my experience, as much as 20% of testing time is consumed by refreshing data between tests. Minimizing the time spent on a data refresh can have a significant effect on how efficiently you complete your testing tasks, especially if you have a particularly large database.

Refreshing test data onto a test system that matches your current configuration can be time-consuming enough, but additional time is often required to resolve the problems that can arise when you are refreshing data onto a new hardware or software configuration. Therefore, you need to plan your data restore strategy carefully, as discussed in my previous article, since failure to do so could delay your test execution.

You may also encounter data corruption as you refresh test data, which can be caused by anything from bad hardware to bad data in the "gold" copy of the database. Earlier, I discussed how to detect possible data corruption by monitoring the database while tests are being run. Data corruption also typically shows up in the system log, which you can inspect using transaction *SM21*.

Once again, proper planning along with a well-formulated escalation plan is the best way to effectively deal with these types of emergencies.

Strategies for Minimizing Data Refresh Times

Here are some ideas for keeping the data refresh window to a minimum:

1. Use locally attached, high-speed interconnections, such as FireWire or SAN switching fabric, to transfer data from your backup device to your database server.

2. Use disk subsystems instead of tape subsystems to store the gold copy of the database that is used for the data refresh. If you must use tape drives, try to implement an unattended restore by using a tape library robot that can automatically mount several tapes.

3. If you have high-end storage subsystems such as EMC Symmetrix, use their data-cloning or data-replication features to create multiple copies of the test data set. You can then simply detach the database server from one disk group and reattach it to another disk group.

Another technique that can dramatically reduce the time required for a data refresh involves using VERITAS Volume Manager with the FlashSnap option. You can have your gold copy of the database on one disk volume (let's call it the "gold volume"), and you can host the working database on two or more additional disk volumes of the SAN (let's call these "disk volume 1" and "disk volume 2"). Once the test run has completed on disk volume 1, causing the database on this volume to become unusable, you simply take it offline and mount disk volume 2 to the database server. You do not have to shut down or reboot your database server, and your next test run can start within a half hour or less! You can resynchronize disk volume 1 using data from the gold volume, independently of the current test run, so that disk volume 1 is ready and waiting for the completion of the test run on disk volume 2.

While this technique keeps the refresh window between test runs to a minimum, it does involve purchasing at least two licenses for VERITAS Volume Manager with the FlashSnap option. However, this software investment is not very large — a setup for two servers (one as the database server, and the other as the data refresh server) should not exceed $10,000 US, depending on the platform involved (contact VERITAS for exact pricing). VERITAS Volume Manager with the FlashSnap option supports almost all of the current SAP operating system platforms, namely Windows 2000/2003, Linux, Solaris, HP-UX, and AIX.

I particularly like this solution because while it works best with a SAN, you can also use a SAN with inexpensive JBOD disks that are SCSI-attached to a server. Another option is to use this solution with high-end storage subsystems such as EMC Symmetrix, Hitachi Data Systems, CLARiiON, and the like. In short, if you already have a SAN, this small investment in VERITAS software will pay for itself many times over, because it will enable you to minimize the idle time of your testing team.

Task #9: Change the Test Environment Between Tests

Some tests may require major modifications of the test environment during testing. For example, in order to compare results with the baseline, it may be necessary to add more RAM or even a completely new server after part of a test has executed. The time required for this change could be considerable — you might require a hardware vendor to come onsite to modify the server, and afterward you will need to run some throwaway tests to properly populate the buffers.

In addition to details on the timing and extent of the change, your test plans should include all the details on who will make the change and the procedures they will follow. For example, the testing team must ensure that the hardware vendor representative understands his or her role and has scheduled time to provide a quick response to your testing team.

With careful planning, you can reduce downtime due to system modifications, but you can't entirely eliminate it. Testing activities will still be interrupted. Plan to use this downtime productively by creating documentation, preparing outlines for your final reports, and so on.

Task #10: Evaluate the Results

Normally, the heavy-duty analysis of your test results takes place in the documentation and presentation phase of testing. Before you close out the actual testing phase, however, you must be certain that your test results are complete, accurate, and meaningful. If there were problems with any tests or with capturing any results, you may have to rerun those particular tests.

The best way to verify that you have complete, accurate, and meaningful results is to carefully review your test plan and evaluate each objective:

- ☑ Did you achieve each goal for your test? This isn't a matter of the actual answer to a question like "Can we get by with one less server?" Rather, it means confirming that you did indeed capture all of the information needed to answer that question.

- ☑ Do you have data for all of your most important SAP transactions?

- ☑ Did you run the tests within all of the parameters specified?

- ☑ Do your results make sense? For example, if you were expecting some performance improvement due to hardware changes, did the improvements recorded during the various test runs seem too large or too small? Or did the improvement in one test run seem dramatically better than the improvement in the other test runs? How can you explain that? If there is a result that you cannot legitimately explain, then you should question it. If the results don't seem reasonable, you should consider retesting.

- ☑ And finally, have you captured or recorded every detail you will need in order to properly document the test?

Be absolutely certain that you have actually achieved what you set out to accomplish before you conclude the testing phase and disband much of your team.

Task #11: Prepare Data for Management Review

In nearly every testing environment I've participated in, senior managers just can't resist asking how the testing is going and what I think the results will be. Remind yourself and anyone interested in interim results that the jury is out until the final reports are presented. It is dangerous to give reports, even

Figure 5 *Comparison of the Top 10 Transactions*

Microsoft Excel - TestResultsOnProjectRedCloud

Top 10 Deemed as Critical OLTP Results under 100% stress

	BASELINE		PRODUCTION		TEST 1		TEST 2		TEST 3		TEST 4	
Transaction	Response Time	Dialog Steps	Response Time	Dialog Steps	Response Time	Dialog Steps	Response Time	Dialog Steps	Response Time	Dialog Steps	Response Time	Dialog Steps
MMBE	0.61	628	0.11	120	0.08	407	0.33	840	0.05	840	0.04	83
ZA21	2.88	225	0.81	744	0.47	4,563	2.43	4,067	1.28	4,500	1.16	4,50
VA01	0.43	1,254	0.6	10,698	0.43	6,181	1.08	5,848	0.37	8,057	0.36	7,49
VA02	0.67	2,766	0.34	2,422	0.26	3,155	0.76	1,037	0.76	1,037	0.21	3,61
VA03	0.32	832	0.37	94	0.23	603	0.6	1,753	0.25	2,277	0.27	2,28
ZH04	0.78	454	0.08	246	0.08	382	1.63	400	0.24	400	0.08	37
ZX11	1.58	677	17.66	8	8.03	116	5.08	480	2.29	480	2.13	48
XZ03	1.28	63	20.07	180	1.37	19	34.71	450	16.26	450	7.27	45
ZL24	16.28	74	19.6	12	22.59	7	0.42	7	0.21	7	0.25	
YC_231	0.13	75	0.04	16	0.4	27	2.96	48	0.25	34	0.42	

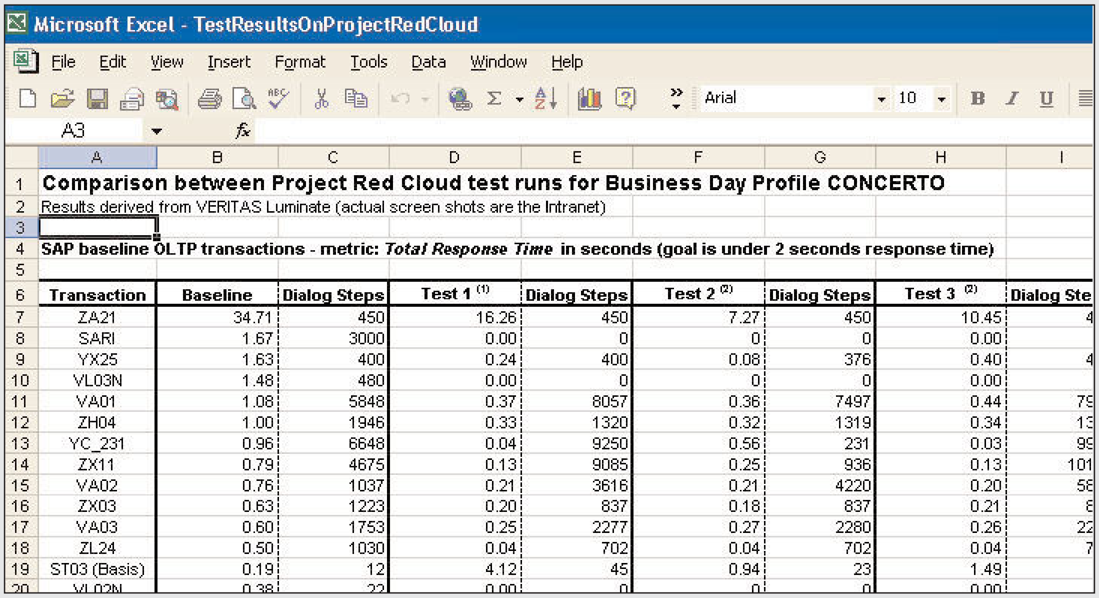
informal verbal ones, about the test results before all tests are complete for just that reason — the results are incomplete! You could very easily present a misleading picture, which then discredits your final test results. Even if your tests are not particularly top secret, stress to your team that allowing non-team members sneak peeks at the results is not helpful to anyone. You'll provide complete results and documentation when the test is complete, but not a minute sooner.

When it comes time to present the test results, remember that your tests were undertaken to answer various business and technical questions. Your tests were undoubtedly complex, but stay focused on why you conducted these tests. Remember that a presentation to management is different from test documentation. Your documentation is very detailed; it includes every description of every element involved in the test, details your methodology, and contains lengthy write-ups, charts, and graphs. By contrast, your presentation to management should be clear and simple — senior managers want answers, and they don't have a lot of time to wade through the details. I always provide the presentation and the test documentation separately, so that managers can read the documentation if they choose, but they aren't required to wade through it. I put the heavy-duty technical details in

✔ *Experience Teaches…*

In one project I was involved in, a junior test run analyst was so taken with several impressive test runs that yielded much better results than expected that, without consulting anyone, this person informed the IT director of those great results. The director then came to the hasty conclusion that more servers were unnecessary and, faced with a deadline to submit his next year's budget, he cut some planned hardware expenditures. Later, after a careful review was undertaken to explain the "better" performance, it turned out that the team had forgotten to start several batch jobs. Had those jobs run, the results would have been far less exhilarating. However, the damage was done, and undoing it became a political exercise.

Figure 6 *Comparison of Key Transactions*



Microsoft Excel - TestResultsOnProjectRedCloud

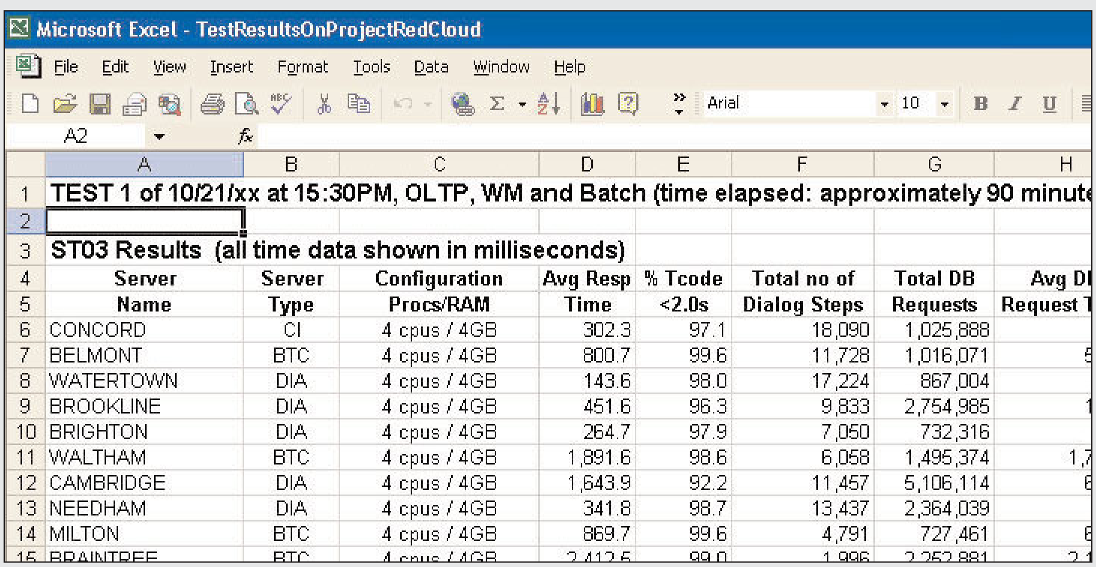
Comparison between Project Red Cloud test runs for Business Day Profile CONCERTO

Results derived from VERITAS Luminate (actual screen shots are the Intranet)

SAP baseline OLTP transactions - metric: *Total Response Time* in seconds (goal is under 2 seconds response time)

Transaction	Baseline	Dialog Steps	Test 1 (1)	Dialog Steps	Test 2 (2)	Dialog Steps	Test 3 (2)	Dialog Ste
ZA21	34.71	450	16.26	450	7.27	450	10.45	4
SARI	1.67	3000	0.00	0	0	0	0.00	
YX25	1.63	400	0.24	400	0.08	376	0.40	4
VL03N	1.48	480	0.00	0	0	0	0.00	
VA01	1.08	5848	0.37	8057	0.36	7497	0.44	79
ZH04	1.00	1946	0.33	1320	0.32	1319	0.34	13
YC_231	0.96	6648	0.04	9250	0.56	231	0.03	99
ZX11	0.79	4675	0.13	9085	0.25	936	0.13	101
VA02	0.76	1037	0.21	3616	0.21	4220	0.20	58
ZX03	0.63	1223	0.20	837	0.18	837	0.21	8
VA03	0.60	1753	0.25	2277	0.27	2280	0.26	22
ZL24	0.50	1030	0.04	702	0.04	702	0.04	7
ST03 (Basis)	0.19	12	4.12	45	0.94	23	1.49	
VL02N	0.38	22	0.00	0	0	0	0.00	

Figure 7 *Results for Test 1*



Microsoft Excel - TestResultsOnProjectRedCloud

TEST 1 of 10/21/xx at 15:30PM, OLTP, WM and Batch (time elapsed: approximately 90 minute

ST03 Results (all time data shown in milliseconds)

Server Name	Server Type	Configuration Procs/RAM	Avg Resp Time	% Tcode <2.0s	Total no of Dialog Steps	Total DB Requests	Avg DI Request T
CONCORD	CI	4 cpus / 4GB	302.3	97.1	18,090	1,025,888	
BELMONT	BTC	4 cpus / 4GB	800.7	99.6	11,728	1,016,071	5
WATERTOWN	DIA	4 cpus / 4GB	143.6	98.0	17,224	867,004	
BROOKLINE	DIA	4 cpus / 4GB	451.6	96.3	9,833	2,754,985	1
BRIGHTON	DIA	4 cpus / 4GB	264.7	97.9	7,050	732,316	
WALTHAM	BTC	4 cpus / 4GB	1,891.6	98.6	6,058	1,495,374	1,7
CAMBRIDGE	DIA	4 cpus / 4GB	1,643.9	92.2	11,457	5,106,114	6
NEEDHAM	DIA	4 cpus / 4GB	341.8	98.7	13,437	2,364,039	
MILTON	BTC	4 cpus / 4GB	869.7	99.6	4,791	727,461	6
BRAINTREE	BTC	4 cpus / 4GB	2,412.5	99.0	1,996	2,252,881	2,1

an attachment that is a simplified version of the test documentation.

Figure 5, **Figure 6**, and **Figure 7** are examples of technical supporting data that might be attached

Figure 8 *"Plain-English" Summary Report*

Project Red Cloud Stress Test Summary

- Best case — Key transactions ran 4-6X faster in the various test runs than the baseline stress test data
- 94.9% of all transactions responded within 2 seconds in all test configurations
  - ❖ Compared to baseline configuration, which delivered 90% of all transactions within 2 seconds
  - ❖ Refer to the intranet for performance details
- Test Scenarios 3 and 4 provided the best overall performance
  - – Test 3 included more RAM on the DB server LEXINGTON
  - – Test 4 included a combined DB/CI configuration
- Highest average CPU utilization was 35%
- 8 GB of memory was more than adequate for the DB/CI combination, as there was almost no paging activity

Company Confidential

✓ ***Tip***

Microsoft PowerPoint is a useful tool for presenting your results, even if you won't be making a formal presentation to a roomful of executives. In addition to being easy to use and having the ability to handle a wide variety of data formats, it helps enforce a "headline and bullets" approach, which is one of the simplest and clearest ways to present results.

to your report. **Figure 8** is an excerpt from an example "plain-English" report in Microsoft PowerPoint form.

When the News Is Bad

What if senior management is not going to like what your tests have revealed? For example, what if the results indicate that your company will be forced to spend millions of dollars just to maintain present performance levels? Or, perhaps your tests show that there is no way your present system can handle online orders sent over your Internet portal, which means the strategic plans of your sales and marketing departments need to be completely overhauled.

If there is bad news, I recommend taking a matter-of-fact approach and delivering the news without editorial comment. If you have recommendations on how to remedy the situation, do not hesitate to proffer them, but do insist on another session to more thoroughly review all the alternatives.

Here are some other strategies I've found helpful in managing bad news:

- ☑ Be even more scrupulous than usual about documenting everything. Because no one likes bad news, the first reaction will be disbelief — the tests must be wrong. You will need to defend your tests and procedures without actually appearing defensive. The best way to do this is with solid documentation that supports the results.

- ☑ Consider carefully which members of your team are skilled at handling potentially volatile situations, and involve them in the presentation phases. For example, if someone is highly respected for his or her technical knowledge, and is a good and persuasive speaker, then bring that person along. A techie who gets defensive quickly or who talks down to senior management because they are "not technical enough to understand" is a sure recipe for disaster.

- ☑ Quantify the bad news as much as possible within the scope of your tests and expertise. If the tests indicate you'll need three new servers, be prepared with details on what size and type of server and the approximate cost. Think through the questions senior managers are likely to ask, and have your answers ready in advance.

- ☑ If possible, present the data as a win for the company. Whatever the tests reveal, focus on the business benefits of the testing itself and the business implications of the test results. Don't mislead as you report, but definitely take the "glass is half full" approach! For example, if results indicate that a significant acquisition of application servers and support infrastructure will be needed to handle the anticipated increase in orders from the Internet portal, you might say, "We can increase our order volume 100% merely by adding 50% more processing capabilities." Taken in this context, the testing project yields positive news regarding the available headroom of your SAP system design.

> ✓ ***Remember!***
>
> *While testing is not a political exercise by nature, it is used to answer business questions, and business is always political. Create a good impression of your testing proficiency, and you are much more likely to be rewarded with budgets that are adequate for "keeping up the good work."*

Conclusion

Your SAP testing must be technically flawless, or as near to it as humanly possible. That is the only way to get trustworthy results on which to confidently base business decisions. The wisdom I have gleaned over years of involvement in SAP performance testing can be distilled into these five basic principles, which, if followed, are the surest route to a technically flawless project:

- ☑ Be sure your team is comprised of the best and brightest people you can recruit. At every stage of performance testing, it's often the people involved that make or break a successful test.

- ☑ Nurture teamwork, and work hard at effective communication. Make sure everyone knows the business purposes of the tests and how their roles support those business purposes.

- ☑ Build support — from your technical experts, to your on-call specialists, to the highest levels of management.

- ☑ Minimize frustrations by preparing your team for problems and having well-established contingency plans. Don't waste anyone's time.

- ☑ And finally, don't lose steam when the tests are complete — you're not finished until you effectively communicate the results to management.

I wish you the best of luck in successfully executing your performance-testing project — though the closer you adhere to the strategies laid out in this two-part article series, the less you will have to rely on luck!

Tim K. Nguyen is Managing Principal of Orchid Consulting Group, Inc., a consultancy that focuses on SAP and e-Business systems performance and business continuance issues and solutions. With over 20 years of experience building successful IT professional services practices, he oversees most of Orchid Consulting's projects worldwide. He regularly conducts SAP knowledge transfer and guides SAP market strategy for leading product and service providers, in addition to performing his hands-on technical work. A graduate of the University of Texas at Austin, Tim has authored two computer-related books published by Simon & Schuster and QUE Corporation, as well as numerous articles in publications such as Computer Reseller News, PC Week, and Lotus Magazine. He is fluent in English and French, and is also comfortable working in Italian. Tim can be reached at tim@orchidgroup.com.

