Continuous Performance Validation Using Modeling

While the use of automated release management tools like Jenkins® combined with automated functional regression testing has been widely adopted, performance testing is not usually part of this automated release procedure.

As a result, changes that may degrade performance may be introduced without notice. Implementing automated performance modeling measurements as part of the automated build eliminates this risk.

This paper details a general approach for *Continuous Performance Validation using Modeling* which can be easily added to an automated build process. It documents an implementation that leverages MJA Technology's VDAB^{tm} to measure and analyze the performance results.

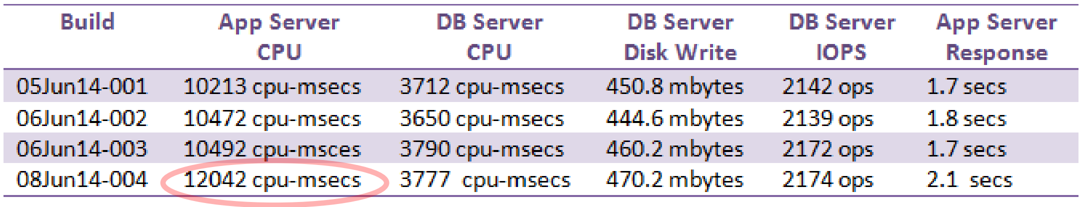
The Approach

The basic approach to calculating a performance model includes the following steps:

1. Enable appropriate performance monitors and log the data at a fairly short interval. (Every 5- 10 seconds)
2. Record the time at the start of applying a standard load.
3. Apply a *standard load* to the system which involves executing a standard set of automated functional tests either through the UI or directly using scripting.
4. Record the time once the load is completed.
5. Calculate the system resources used during the test.
6. Compare resource costs to the previous release and review the release trends.

While the approach is straightforward and can be done manually using a spreadsheet, automation ensures that it will be done consistently with each release.

While many metrics can be used to determine the performance of a system, some of the key measurements are shown in the sample table below. This table displays the values obtained during four different builds:



Build	App Server CPU	DB Server CPU	DB Server Disk Write	DB Server IOPS	App Server Response
05Jun14-001	10213 cpu-msecs	3712 cpu-msecs	450.8 mbytes	2142 ops	1.7 secs
06Jun14-002	10472 cpu-msecs	3650 cpu-msecs	444.6 mbytes	2139 ops	1.8 secs
06Jun14-003	10492 cpu-msces	3790 cpu-msecs	460.2 mbytes	2172 ops	1.7 secs
08Jun14-004	12042 cpu-msecs	3777 cpu-msecs	470.2 mbytes	2174 ops	2.1 secs

This illustrates why the measurement of the actual resource cost is valuable. While the test illustrated an increase in average response time in the release on June 8th to 2.1 seconds, this measurement alone would not conclusively identify the problem and would not identify the source and magnitude of the problem. The significant increase in the amount of CPU used on the application server (up almost 20% to 12042 cpu-msecs) provides a) clear supporting evidence for the problem, b) identifies the limiting resource and c) allows estimation of its expected impact on the overall capacity of the system.

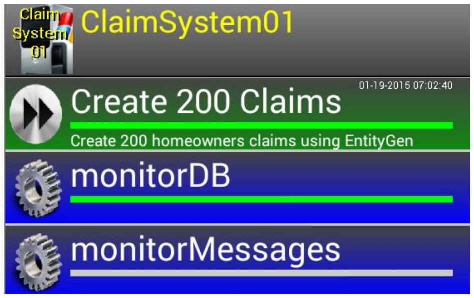
The promotion of this build could be delayed till the performance issues are resolved and retested.

Case Study – Performance Regression for Guidewire ClaimCenter

The following section illustrates automated performance regression implemented for a Guidewire ClaimCenter® project using the VDAB^{tm} performance monitoring and modeling tools. While different standard loads can be used, generating a specific number of typical claims represents a useful characteristic load for a claims system.

While the screens show the procedures initiated by single click on the VDAB UI, **this same procedure is typically invoked using a call from Jenkins or other automated release tool and is completely automated**:

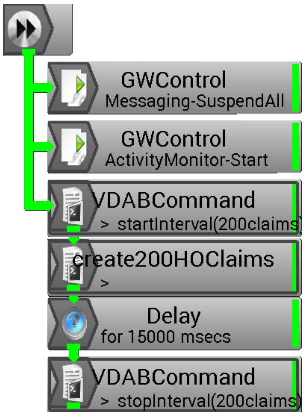
Test Control using VDAB



The screen shows a VDAB view of the monitoring and test control flows or programs that are running on the test system.

A number of monitoring programs are running to acquire the measurements needed for the modeling.

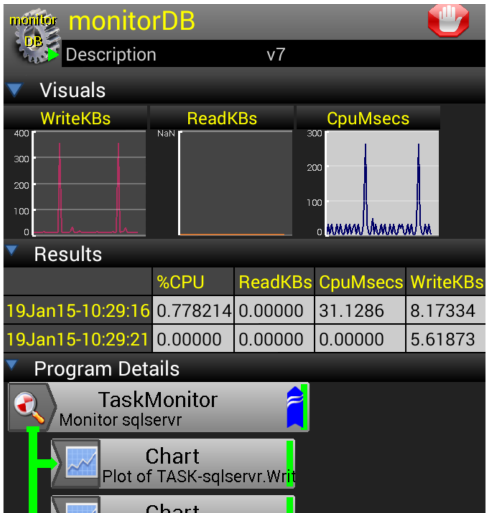
Clicking on the program titled *Create 200 Claims* executes the regression test and calculates the results



When clicked or invoked from a script the *Create 200 Claims* program actually turns off messaging to control the load on the system, declares a test interval called *200claims* and calls Guidewire's EntityGen tool from the command line which causes 200 homeowner claims to be created.

After the load is completed it waits 15 seconds and then declares the interval over.

Monitoring Using VDAB



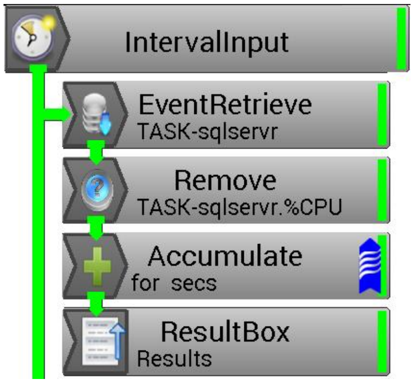
	%CPU	ReadKBs	CpuMsecs	WriteKBs
19Jan15-10:29:16	0.778214	0.00000	31.1286	8.17334
19Jan15-10:29:21	0.00000	0.00000	0.00000	5.61873

Several monitoring program are running during the test and acquire all the data needed to complete the model.

One of these programs, *monitorDB*, monitors the key metrics for the SQL Server database and saves the data so that it can be used later to calculate the performance costs.

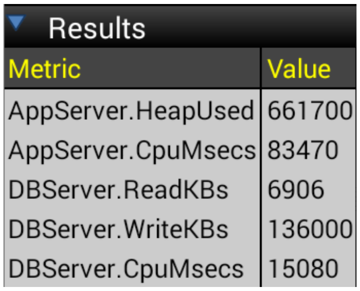
Other programs monitor the messaging backlog, app server CPU, app server heap, and the number and type of business events

Calculating the Test Results



After the test is completed, the results of the test are accumulated for the length of the test and written to both a visual window and to the database for trend analysis.

The modeling steps at the left are just some of steps automatically invoked when the test is completed.



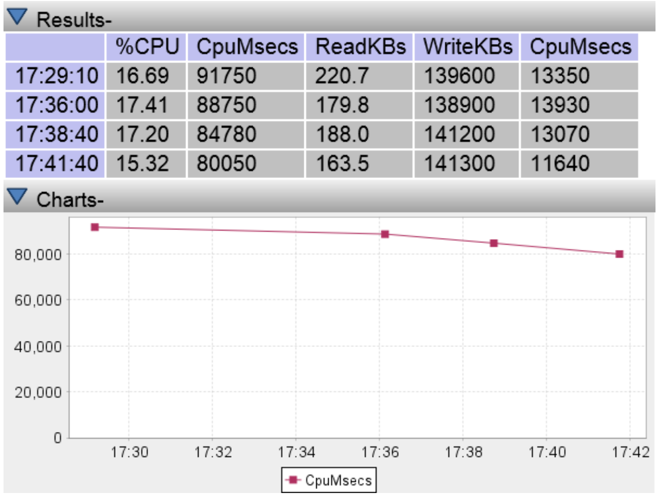
Results

Metric	Value
AppServer.HeapUsed	661700
AppServer.CpuMsecs	83470
DBServer.ReadKBs	6906
DBServer.WriteKBs	136000
DBServer.CpuMsecs	15080

The results of the data are displayed on the VDAB UI for immediate review.

The results are also saved to a database so they can be used for later trend analysis

Monitoring the Trends



Results-

	%CPU	CpuMsecs	ReadKBs	WriteKBs	CpuMsecs
17:29:10	16.69	91750	220.7	139600	13350
17:36:00	17.41	88750	179.8	138900	13930
17:38:40	17.20	84780	188.0	141200	13070
17:41:40	15.32	80050	163.5	141300	11640

Charts-

With VDAB, trend tables and graphs are immediately accessible using a web dashboard.

Performance issues can be identified for the latest release.

Project performance trends can be observed.

Implementing Automated Performance Modeling

Adding performance modeling to your automated build using VDAB is a simple project that can generally be completed in less than one week. The following table details the activities that are typically required:

Hours*	Activity Description
8	Identify one or more standard loads that are part of the current automated regression
4	Implement callouts from the automated build tools or build scripts
6	Install VDAB to monitor the desired test environment
4	Test monitoring agents and the VDAB installation
8	Add customized trend monitoring dashboard links to the production dashboard

*This implementation effort is typical and will vary depending on the system and the number and types of metrics monitored.

For companies that would like to get assistance in deploying VDAB modeling, MJA Technology offers a focused service quickly completing the entire installation.

Contact us for a VDAB modeling demo or a no obligation initial consultation.