

Performance Test Automation within IBM Tivoli PACO

Andrew Rindos, Steve Woolet, Joseph L. Carter, Carol Ames, Xuemei Wu, Yasutaka Hirasawa
and Rupinder Gill
IBM Tivoli, P.O. Box 12195, 3039 Cornwallis Rd., RTP, NC 27709
Email: rindos@us.ibm.com

IBM™Tivoli™ PACO (Performance, Availability, Configuration and Operations) products represent much of the IBM on-demand infrastructure/middleware and include: IBM Tivoli NetView™, various ITM (IBM Tivoli Monitoring) products, IBM TCM (Tivoli Configuration Manager), IBM TEC (Tivoli Enterprise Console™), IBM TBSM (Tivoli Business Systems Manager), TSLA (Tivoli Service Level Advisor) and many others. The performance evaluation support of these products was recently consolidated into one department, and its role was expanded to include: formal performance modeling in order to provide the necessary high and low level design recommendations to development, "global" (system-level) performance tuning guidelines, and more effective capacity planning tools and recommendations for sales/customers.

It was recognized that an order of magnitude increase in the amount of data collected was necessary, implying that tests needed to be run day and night, as well as over weekends and holidays. The level of automation within PACO performance testing therefore needed to be dramatically and uniformly increased, and standardized across the organization. The additional benefits of automation included: increased coverage of performance-related product defects; improved statistical significance and repeatability of results (with larger sample sizes); reduction of errors due to complex repetitive setups and test processes; applicability to frequent build and regression testing (largest source of lost test time: setting up product tests for builds that do not work); and more time available to the performance team to do other work.

A dedicated team was defined that was purely responsible for the development and maintenance of test automation code, and met on a regular basis (with one another

and with designated automation contacts for each product test team). Automation was done in parallel to existing test duties and schedules.

It was found that most tests could be broken down into a shared atomic set of subprocesses: test system setup and initialization; input load creation; input load generation; parameterized test control; data collection; data reduction and analysis; etc. For many of these subprocesses, the basic structure was the same, regardless of the actual product examined, and so standard parameterized coded procedures could be defined, with modifiable interfaces for the different products. Other subprocesses were unique to the product and required more specialized code - but all could be imbedded into a larger control program consisting of shared code modules.

The figure below summarizes the existing (standardized) automated test paradigm that has been developed for all PACO products. All performance tests parameters were catalogued, in order to define a standardized test control file structure (for the **.ini file** below). This control file is used by a tester to define the parameter variations (for $i = 1$ to n , for $j = 1$ to m , etc.) that in turn define a series of automated performance test runs that the tester desires. (The **.ini file**, as well as the control program, PERL modules, etc. below, reside on a controller machine, that ideally is separate from the test client and server machines executing the actual performance tests. However, all control programs could reside on one of the test subject machines.)

The parameter values in the **.ini file** are fed to a control program (**TestDrive.pl** below, written in PERL) that calls up the appropriate **PERL Modules (pm files)**, e.g., **TestUtils.pm**, **Tec.pm**, etc. (see below), some of which may be product specific.

The pm files issue the appropriate commands to either a **STAF (Software Test Automation Framework) agent** or to **Rational Test Manager**. STAF is a standardized automation framework (see www.staf.org) that can be installed on a wide variety of operating systems to remotely start and stop processes, programs, etc., and otherwise control a series of automated tests distributed across several machines. Test Manager is the centralized control program for **Robot** and **XDE Tester**. Robot is used to emulate large numbers of web consoles making requests against an application server. XDE Tester is used when the actual application itself, e.g., a Java console, needs to be running when generating requests. In these cases, the client application is itself the target of the performance studies.

The STAF agent on the controller machine communicates with the corresponding STAF agents on the remote test machines, initializing the machines as necessary, starting up various performance monitoring

programs (e.g., Perfmon on Windows machines, PTX on AIX machines, etc.), generating test event loads, etc. - i.e., issuing all appropriate commands to start, control and terminate the tests and collect the resultant data. Test Manager similarly communicates with its Rational agents on the test client machines to generate appropriate (real or emulated) web and/or Java console requests against one or more designated application servers. TSTR opens a PCOM session on the controller machine, through which control commands can be passed to a mainframe running z/OS or S/390. (STAF supports the zLinux platform.)

All data is collected into standard output test files, that are rolled up to the controller machine. The format of the data output files have been standardized, so that various programs can analyze and process the data, generate standard graphs and reports, etc.

™ Trademarks of International Business Machines Corp. in the US/other countries.

