Workshop on Performance and Reliability -- WOPR9

Theme: Pushing the Boundaries of Performance Testing Tools

Experience Report: Load Testing a 3-Channel EDI & MQ Messaging System

Dan Downing / Microsoft / Redmond WA – September 27, 2007

Abstract

Performance testing complex systems can push the boundaries of any single tool, no matter how broad and deep its capabilities may be. We expected to push our tools' boundaries as we tackled benchmarking of a complex B2B system through which our customer enables 100+ retailers and 4000+ retail vendors to exchange data on 120 million products in near real-time, in what is called a trading partner "value-added network".

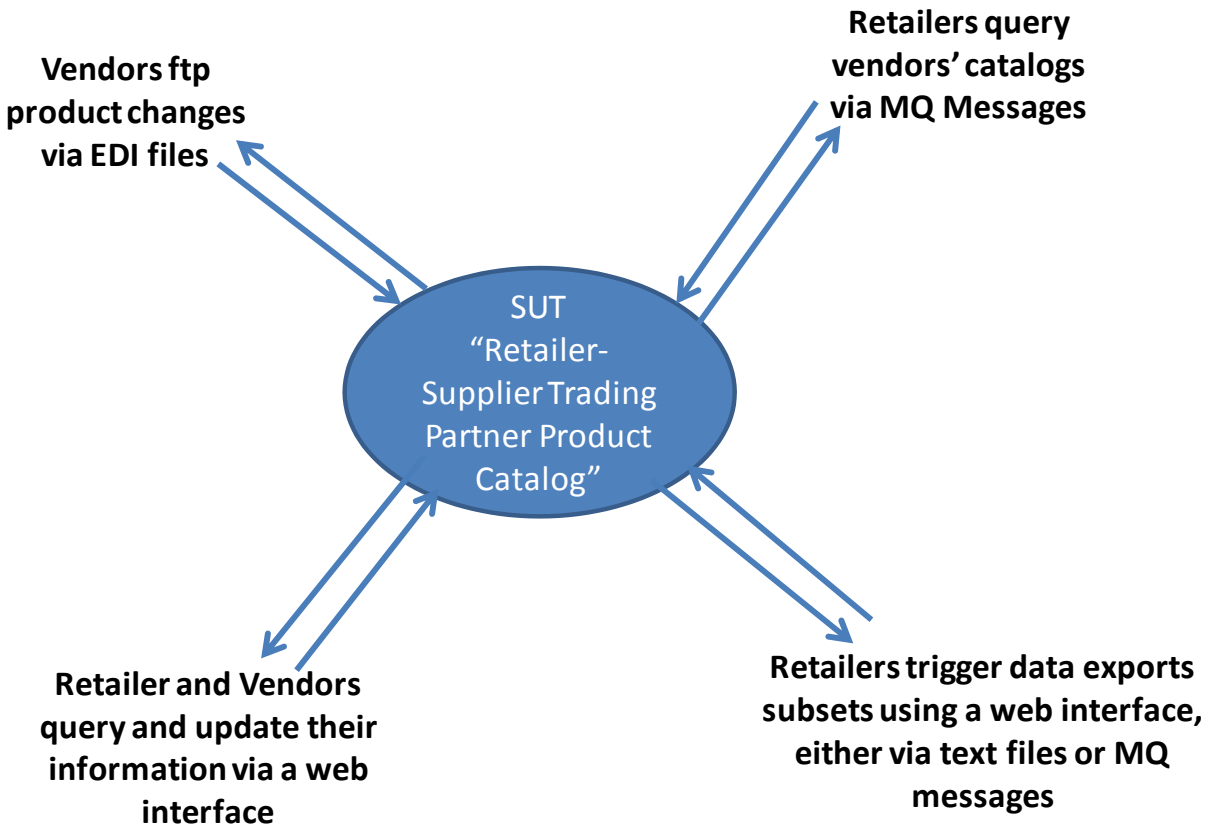
Retailers and Vendors access the system via three interfaces – EDI files that are ftp'd and processed in "batch", XML messages that are transmitted to IBM MQ-series messaging servers and processed in real-time, and web browser transactions that either update individual product records or trigger data exports that are either downloaded as text files or transmitted as MQ messages.

Performance-testing this system would require exercising all three interfaces simultaneously: Transmitting of large of volumes of files through both the EDI and MQ channels, and simulating http transactions through the web interface. An initial assessment gave us confidence that LoadRunner could handle the load delivery, but the solution would require a little-known LoadRunner MQ add-in and perl scripting to pre and post process production files and capture execution times from server logs.

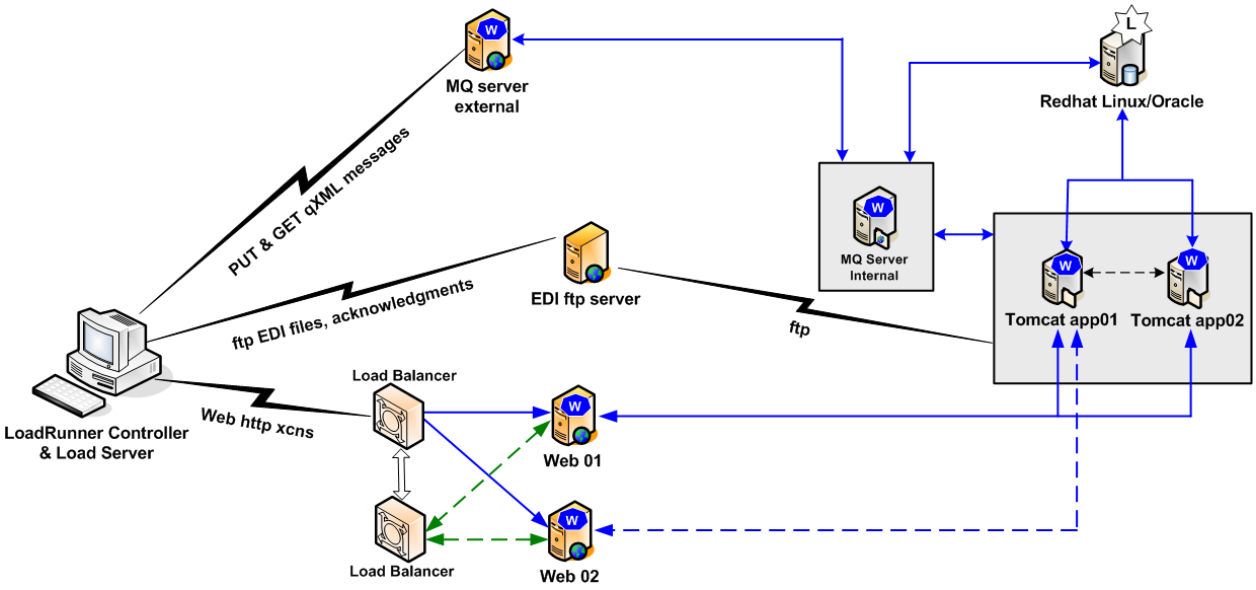
Of course, like with any challenging project, at the onset we did not know what we did not know. This is the story of the challenges encountered and solved in this project.

The following pages are presented as diagrams and talking points used to tell this story to the 25 WOPR9 attendees.

1. **The System**


Vendors ftp product changes via EDI files
Retailers query vendors' catalogs via MQ Messages
SUT "Retailer-Supplier Trading Partner Product Catalog"
Retailer and Vendors query and update their information via a web interface
Retailers trigger data exports subsets using a web interface, either via text files or MQ messages

2. Architecture and communication protocols



3. Our process and its challenges

Step	Challenge
Define goals	Persuade Retailers to migrate from mature legacy to new environment; repeatable by QA
Quantify performance targets	Mix of production transaction rates, stand-alone synthetic transaction benchmarks
Collect & adapt data files	Production files not usable in test env. 'as is'
Count transactions 'inside' the files	Variable no. of xcns in each file
Flood the channels	Not single interactive xcns, but files in volume
Time the async message round-trips	Xcns not synchronous; need to match PUT and GET message times
Monitor system resources	Mix of Windows, Linux server & services
Compute transaction rates	Deeper granularity than typical load test results yield by default
Summarize, analyze, present results	The usual reporting challenges

4. The MQ Solution

Step	Process	Tools
Data & Count	Harvest prod files, update Message block, sort by target DB, count Function blocks; Prod DB snapshot; change Retailer MQ config	Perl script (update key fields, count transactions, sort by target DB) SQL scripts (change Retailer MQ configs)
Flood	Transmit files to multiple MQ server queues quickly	LR w/ MQ protocol add-in, *MQ Accelerator* scripts, custom coding
Time	Capture response messages, correlate msg IDs, compute (response msg time) – (xmit msg time)	Intelligent *MQ Accelerator* scripts
Monitor	Monitor MQ, java middleware, DB server resources	LR monitors for Windows & Unix Server; Sun Jconsole for Tomcat
TPS	Compute tps in 1-minute average buckets, graph	Excel model
Report	Present key observations, conclusions & recommendations on throughput and system capacity	Word

Pushing the Limits: LoadRunner for MQ + perl

- **Context:**
  - Previous testing done by developers using home-grown java harness that created synthetic transactions fed to the java server (by-passing MQ tier)
- **Limit:**
  - MQTester is 3rd party developed LR add-in, little-understood support step-child; requires MQ expertise
- **Solution:**
  - LR with MQTester add-in to replace java harness
  - MQ consultant for MQ expertise
  - Proprietary "MQ Accelerator" script set as scripting jump-start
  - Extend Accelerator to handle application's requirements (message file sets, 'dynamic queues')
  - perl script to adapt Prod files, count transactions inside qxml files, sort into target DB

5. The EDI Solution

Step	Process	Tools
Data & Count	Harvest prod files, count transactions; Prod DB snapshot	Perl script (count xcns)
Flood	Transmit files to ftp mailbox quickly	LR ftp protocol scripts
Time	Capture acknowledgement file, match file IDs, compute (acknowl. time) – (ftp time)	Excel model
Monitor	Monitor MQ, java middleware, DB resources	LR monitors, Jconsole
TPS	Compute tps in 1-minute average buckets, graph	Excel model
Report	Present key observations, conclusions & recommendations on throughput and system capacity	Word

Pushing the Limits: LoadRunner & EDI

- **Context:**
  - Previous testing done by developers using home-grown java harness, synthetic transactions fed to the java server (by-passing EDI tier)
- **Limit:**
  - ftp scripts offer no automatic way to match ftp'd EDI file w/ acknowledgement file to determine round-trip time
- **Solution:**
  - perl script to count transactions inside EDI files
  - LR ftp script to ftp files, obtain 'acknowledgement' arrival times
  - perl script for matching 'acknowledgement files' to source files
  - Excel to summarize and graph results

6. Conclusion

Creative test design, careful tools selection, recruiting an MQ expert, lots of custom C coding in LoadRunner, and creative application of perl, has enabled us to overcome the challenges of this project so far – it is still 'an experience' in process.

What at first seemed unlikely to be solvable as a LoadRunner solution which we could turn over to the customer's QA team, is looking doable. Stay tuned for the conclusion!