WOPR8 Experience Report
Critical Incidents in Performance Testing
Dan Downing – April 7, 2007

Abstract
Having participated in over 100 performance tests for more than 75 companies over the last 8 years, spanning in complexity from small web retail sites to enterprise ERP systems, I have experienced critical incidents in all four category – process, tools, teams, and stakeholder engagement. Each critical incident taught an important lesson that has raised the quality of our service and the performance IQ of our testing team.

In this experience report, I will present two salient critical incidents that should provide other WOPR attendees with specific techniques in modeling with end-user participation, tool vetting, understanding underlying technology, and results reporting.

Lawsuit Anyone?
In October of 2000 I spent two weeks onsite at Kmart's Blueghligth.com space across from Fisherman's Wharf in San Francisco, with one of my senior performance engineers, stress testing their Martha Stewart branded web store in preparation for the holiday season. Keith and I were embedded with their development and QA teams in a fishbowl conference room, working 12+ hours a day, modeling and testing the typical mix of ecommerce transactions (browse, search, add-to-cart, checkout, register, etc.), with LoadRunner.

It was a dynamically evolving site, as the business owners kept tuning their store model, the "creatives" kept changing section designs, developers kept modifying and fixing code, and the infrastructure team scrambled to build the production infrastructure hosted at a Bay Area hosting facility about an hour away. Despite the long hours and the constant re-recording and tweaking of our LoadRunner scripts, it was a frenetic but infectious "open bullpen" environment, with 24/7 free M&M's, soft drinks, coffee, and food, and where Martha herself could often be seen on the floor chatting it up with the business team.

We ran well over a dozen tests up to 4000 concurrent users, driving load from Mercury's hosted load injectors in Santa Clara, and from our own injectors in Memphis. When the Web Director declared that site performance was "good enough" in early October as they went into the final critical activities to go live, I asked him details about the software version and the environment we tested on for our final report. He brushed us off, saying that the graphs and interim results we'd already generated were good enough. With that we packed our bags, high-fived the development team, and left.

Fast-forward to the day after Thanksgiving: Several large sites, including Amazon, BestBuy, PayPal, and, yes, BlueLight had experienced performance hell during the start of the shopping onslaught. Our CEO was getting calls from BlueLight, and from the media, and we were suddenly on the defensive.

Our CEO called a big meeting that afternoon: "What had happened? Hadn't we validated their site for 4000 concurrent users (no, we never got that high)? The Web Director was pressing us. What facts did we have to respond with?"

I knew that we'd left on October 2nd, when the Director had said we'd done enough. I knew their application was still undergoing changes, and would be right up to go-live in late October. I had spreadsheets with logs of the 18 test sessions we'd conducted. I had tons of graphs, status reports identifying issues we'd found (both with our own test environment as well as with their site). I knew that the ATG Dynamo performance expert was still on-site after we left trying to fix the performance-halting garbage collection 'heartbeat' problem we'd identified.

But I could not provide: the version of the release we'd last tested (if there even was one); the configuration of the environment we had last tested on; the extent to which the final product catalog had been loaded into the database at the time of testing, or even the size of the database.

Net-net: I could not produce the report we'd presented to the customer—because we hadn't. I'd allowed the brash, impatient director to talk me out of producing it—which would have forced me to drag the aforementioned specifics out of the systems guy, the DBA, the Sr. Developer.

The following Monday, December 4, InformationWeek came out with a front-page article on e-retailing, featuring none other than the Web Director at BlueLight, a quote about having used AppGenesys (the company I worked for at the time) to do their stress testing, an admission that page response times had gone up to 30 seconds during the peak, and a defensive quote from *our* CEO back-peddling about the "unpredictability of the web, and all the factors you can't control".

Thanks to the frenetic pace of the time, some fast talking by our CEO, and providence, we escaped a lawsuit, any *really* bad press, and no one got fired. But for me it was then – and is still now – my very first (certainly not my last) *critical incident* in performance testing.

Since then:
- I always generate a final report – whether the customer values this report or not. And I find away to present it, even if only via email (which I carefully file).
- I document exactly what we tested: the software release, the business processes, the configuration of the system-under-test
- I note the date and times of our "best and final" test
- I describe how and what we tested, including any limitations or simplifying assumptions we were forced to make
- Key findings, conclusions and recommendations are made with carefully chosen language, being careful to not overstate, be overly-optimistic, use overly-positive language, or assert 100% confidence in our results.

(Click here for [sample report](#) and [InformatioTwo salient critical nWeek article](#)).

Know Thy Test Tools

Ever been caught in embarrassing situations with your test tools about your knees? Well I have. Twice.

10,000 User Test with OpenSTA.
A company that develops and markets and an on-line student testing software and infrastructure package was being evaluated for scalability by the State of Oregon—which if accepted would result in a major purchase. We were contracted through one of our partners to do the testing. I'd never run this size test, not even with LoadRunner, and OpenSTA was the only tool we could use (based on licensing costs and our experience). I said "sure, we can do that!" Oh by the way, up to this point to biggest test we'd ever run with OpenSTA was 1500 users…

The test was to simulate 10,000 students logging in under unique IDs, answering 35 questions, and logging out, with an end-to-end duration of 50 to 60 minutes, including think-time. We would launch users at a rate of 5 per second, ramping up to 10K over a 35 minute period. The testing would occur on the company's production environment, which required scheduling of each test at night, and substantial environment preparation by the customer for each test.

The system-under-test consisted of 28 web servers running Linux and Resin, and a database server running Postgres SQL. We would run two preliminary tests at 2500 and 5000 users before the final test at 10,000—all this in a 2-3 week period starting the week of Thanksgiving (2004).

We kicked off the project two weeks before Thanksgiving. Script development went smoothly, as did a 30-user shakedown test conducted the Tuesday before Thanksgiving. To drive this load, we pressed into

service a big honking server: quad 1.5 GHz Xeon Hyper-threaded Dell 6650 with 4 GB memory, 160GB ATA disk and dual GIG-E network cards. We were ready –and confident--for the 2,500 user test on Wednesday afternoon.

The first test started, users started ramping (at 3 users/second for this test) and everything seemed to be working well, both at our end, and the customer was seeing users logging in. But the good news was short lived—at about 6 minutes into the test, OpenSTA was showing about 900 users running, but network throughput effectively stopped. We and the customer tried to diagnose on the fly—network failure at either end? System event on the load injector? The customer reported "runaway http connections". We stopped the test and agreed to diagnose fully and get back the day after Thanksgiving.

Was it the new load injector? Was the problem with hyper-threading? Was our firewall overloaded? Some issue with OpenSTA? The load injector event log showed nothing. Our perfmon logs revealed no limiting resource. Our networking guys examined our firewall stats and concluded it had plenty of capacity. What to do next? Isolate the variables, research each one, etc. This took a few days…and cost us precious time and credibility.

Our performance engineer Linda dug into the OpenSTA FAQs and forum and came up with a hypothesis: given the high ramp rate, and the script's large number of steps opening a large number of sockets for page resources that were being haphazardly closed by the default DISCONNECT nn statements OpenSTA inserts during recording – we were running out of sockets! That's when we learned about explicitly inserting DISCONNECT ALLs after each page! We conducted an internal test with and without SYNCHRONIZE REQUESTS, running on our hyper-threaded injector, and proved conclusively that this was an issue.

We finally conduct a successful 2500-user test on December 6—gaining back a little customer confidence, and moving us forward to schedule the 5K test—though with internal confidence greatly weakened, and imagining more disasters lurking.

Well guess what limit we ran into next? Yes, the infamous 1664-virtual-user-per-Commander OpenSTA limitation, related to the way it uses file handles and the file handle limit on Windows Server. More egg on face, more time and credibility lost, while we scrambled to cobble together more load injectors, then painstakingly merge Timer results to graph results.

Our customer contact escalated his frustrations to his boss, who then called our partner's CEO, who then called our CEO—it was not pretty. But somehow we managed to keep them from throwing us out (they were as committed as we were by now) – and by January 4 we were able to press seven injectors into service, fielded a team of 5 people to coordinate launching of separate Commanders on each box, with our networking guys monitoring our entire infrastructure, and managed to conduct a successful10K test.

After the test we had do some creative merging and massaging of the huge Timer and HTTP logs, and finally were able to publish a report that showed good scalability at the target load that the customer could present to the State of Oregon. That deal – and our butts—were saved.

Critical Incident # 2: Know your testing tools – their capabilities and limitations – and TEST TEST TEST the capacity and limitations of your load injectors and test environment.

Having learned *this* hard lesson, now ask me about the limitations of using LoadRunner to test a Citrix application. ARGH!