WOPR13 Experience Report
Theme: Performance Rules of Thumb

Title
Rule 1: Apply the Fundamentals! Rule 2: See Rule 1!

Abstract
If I use performance testing rules of thumb, they spring from applying the fundamentals: identifying business objectives, modeling the load, monitoring system resources, interpreting results and removing bottlenecks.

I apply these fundamentals from the first conversation with a customer to gauge probability of success and set expectations on the effort involved. I use the dialog to assess seriousness of intent, project viability, and plan the scope and approach.

In this experience report I will draw on two enterprise application projects to compare and contrast how these fundamentals played out, how the business results compared, and what we can learn as performance testers to successfully assess, plan and execute new projects.

Project #1
Oracle R12 at a national fire and safety service company

Context
- Company was upgrading from Oracle R11 that was plagued with performance problems to R12, a release that had new functionality they needed, and which Oracle promised would have the needed performance enhancements.
- R12 is the first Fusion release from Oracle, introducing new java objects and more efficient client-server interface using sockets rather than servlet
- Some (but not awful) contentiousness between IT and business units.
- Complex application modules, including service scheduling, customer master management, parts inventory, and sales lead management, interfaces to IVR and PC synchronization, user communities in multiple departments and 100+ US sites, number several thousand.
- Distributed LoadRunner load testing environment spanning multiple networks, firewalls, local and remote load injectors, and administrative headaches as a result

Fundamentals
- Conducted 3-day on-site assessment 6 weeks before the start of the project; developed detailed scope and approach plan, including quantification of target load based on analysis of their usage stats
- Held a tightly- coordinated 1-week Discovery on site with a team of 4 people. Meeting with key business users, to drill down on workflows, acceptance criteria, data dependencies, and fine-tune the target load (users and completed transactions)

- Met with IT in advance to define development environment (frozen code), remote access, LAN-LAN vpn from us to them so we could leverage out hosted load injectors, resource monitoring access (firewall ports, server ACLs), configuration of their inside-the-firewall injectors,...
- Worked out the major LR-R12 protocol record/playback “gotchas” during the Discovery (HP tech support actually helped!)

Successes

- Gained quick working relationship credibility with business and IT
- Established a clear test plan – including performance acceptance criteria!
- Met with IT in advance to define development environment (frozen code), remote access, LAN-LAN vpn from us to them so we could leverage out hosted load injectors, resource monitoring access (firewall ports, server ACLs), configuration of their inside-the-firewall injectors,…
- Established a sound plan for a coordinated team effort, divided up by application area, and coordinated data dependencies by mapping out the data flow

Gotchas

- The Dev app was full of bugs! Thus, not really frozen…had to endure blockages, multiple updates, and develop/report issues as encountered and press for fixes
- After our first test, their tuning expert realized they’d implemented R12 with the older/less efficient “servlet” approach, vs. the newer socket approach; upon their changing this, all 20/23 of our scripts were broken; initialization sections had toi be re-recorded
- The Advanced Scheduler functionality was not working until near the end, requiring yet another round of re-scripting
- The distributed LR environment had so many “jurisdictions”, moving parts, flaky components, as to significantly impact productivity, and require careful script version control

Learnings

- Insist up-front on gaining a deeper awareness of the custom development and patching plan to minimize re-scripting impact
- In the next R12 project, dialog with the Oracle DBA up-front about the Oracle client config—again to mini8mize re-scripting
- The ground-breaking we did with LR 8.1.4 and R12 project made for a well-received talk at HP Universe
- Dialog with business users about performance acceptance – they usually know where the really slow steps are!
- Net-net: Even when applying the fundamentals properly, no project is complete until 3 very bad things happen

Illustrations

Environment diagram; target load Excel; benchmark results; a graph or two…

Project #2
Manufacturer to retail trading partner messaging app for a national provider of business community B2B solutions.

Context

- Company sold a complex integrated solution of applications and dedicated infrastructure to a large computer manufacturer that projected traffic volumes that exceeded the known capacity of the apps
- Capacity testing forced by computer manufacturer just weeks before the Holiday peak
- Customer's performance testing environment is home-grown, and anything but robust.
- Test env /= Prod, and known bottleneck (SAN IO) 2 generations older than Prod, so no way to extrapolate Test env results to Prod
- Critical monitoring of app queues done via a slow, impractical UI; no way to measure/log anything about the load or the infrastructure
- Operations-owned initiative, and Ops MO is "crisis mgt"

Fundamentals

- Tried to create clear responsibilities by creating a project charter; was told, "no, we won't be running this like a standard project; key contributors are too busy"
- Developed an approach outline and some sample graphs of the results we would be shooting for, and convinced the project owner to hold a 1-hour planning call with key infrastructure people that could implement some monitoring
- As the "load test advisor", my role was limited -- i.e., I was not running the project – a good thing in this environment; but this also meant that no one had the "architectural understanding" of both the Apps and hardware and what app components ran on what pieces of hw
- The "target load" was difficult to develop—production volume baseline plus certain traffic types projected to double – and challenging to document in a standard way – and I was learning the subject matter through a fire hose on the fly…

Successes

- Gained credibility quickly with project owner with the approach outline and the sample graphs I created – and managed to define a role I could be successful in
- All were convinced of need to monitoring at 15-second granularity, and the need to create better utilities to monitor app queues (it turns out QA was working on these in parallel – but not ready in time)
- SAN IO bottleneck, and corresponding Oracle lock contention, was clearly identified – and valued!

Gotchas

- The Test env failed to support the target load – and when I tried to say "no point in going further", VP said "lower the target and keep testing!

- Developer brought in to review findings – and found that one of the key “slow apps” was not configured in an optimal way; that’s what happens when you’re flying fast!
- Ops VP made pint that “we’re not seeing these backlogs in Prod, so they must be artificially induced by injecting too high a load”. Hard to argue this; as a result, testing has resumed with a reduced load, and is on-going

Learnings

- Not all project “conditions” allow for a full application of the Fundamentals; learn to be flexible, without compromising on the critical items (in this case, refining target load by comparing results to Production)
- Use “mocked up results graphs” to get everyone to “see the value” in what we’re out to measure
- If you don’t /can’t own the project, carve out an advisor role that still adds value

Illustrations

Approach outline; mocked up graphs; actual SAN IOPS graphs; target load sheet