PERFORMANCE ENGINEERING QUICK-START WORKSHEET

Goals

- Determine the stakeholders' needs and expectations.
- Ensure accountability can be assigned for detected performance issues.
- Ensure project complete criteria is agreed upon.

Pre-Project

1. How many users (human and system) need to be able to use the system concurrently.
    a. What is the total user base?
    b. What is the projected acceptance rate?
    c. How are the users distributed across the day/week/month?
    d. Example:
    Assuming evenly distributed billing cycles throughout the month, users spending about 15 min each viewing and paying their bill, and the site is generally accessed between 9AM EST and 6PM PST (15 hours). Calculated concurrent users with the following formula.

    (total monthly users)/(30 days a month * 15 hours a day * 4 {note, 60min/15min per user} = daily average concurrent user load.

    Normally test to 200% of daily average concurrent user load.

    If 1 million monthly users…

    1,000,000/(30*15*4) = 556 concurrent users. (2,222 hourly users)
    Recommend testing up to 1000 concurrent users (4,000 hourly users).

2. General performance objectives
    a. Can service up to a peak of <*????*> customers an hour without losing customers due to performance issues?
    b. System stable and functional under extreme stress conditions?
3. What are the project boundaries, such as:
    a. Are all bottlenecks resolved Or….
    b. Best possible performance in <*????*> months Or…..
    c. Continue tuning until goals are met Or…..
    d. Continue tuning until bottlenecks deemed "unfixable" for current release?

Pre-Testing

1. What are the specific/detailed performance requirements?
    a. User Experience (preferred) - i.e. With 500 concurrent users, a user accessing the site over a LAN connection will experience not more than 5 seconds to display a small or medium bill detail report 95% of the time
    b. Component Metric (not recommended) - i.e. Database server will keep memory usage under 75% during all tested user loads
2. Detailed description of the test and production environments.

    a. Associated risks if environments are not identical
    b. How to best mark up performance if not identical
3. What is the availability of system administrators/developers/architects?
4. What is the test schedule
    a. Can tests be executed during business hours?
    b. Must tests be executed during off hours?
    c. Must tests be executed during weekends?
5. Are system monitoring tools already installed/being used on the systems under test
    a. Perfmeter, Perfmon, Top, Jprobe, PerformaSure
    b. Who will monitor/evaluate monitoring tools?
6. Are any other applications/services running on the systems to be tested.
    a. Think about associated risks to shared environments, like memory, disk I/O, drive space, etc.
7. What are of the types of tests/users/paths desired
    a. Target for 80% of user activity
    b. Always model system intensive activity
8. Based on answers, create Test Strategy Document

Test Design/Execution

1. Design tests to validate requirements
    a. Create User Experience Tests
    b. Generate loads and collect Component Metric data while under load.
2. Always Benchmark application in a known environment first
    a. Architecture need not match
    b. Benchmark tests do not need to follow user community model exactly
    c. Benchmark tests should represent about 15% of the expected peak load
    d. Look for problems "low hanging fruit"
    e. Do not focus on maximizing User Experience, just look for show stopping bottlenecks
3. Benchmark in Prod environment if possible
    a. Look for problems with code/implementation
    b. Look for problems out of scope that client must fix to meet performance goals (i.e. Network, Architecture, Security, Firewalls)
    c. This benchmark must be identical to the benchmark conducted in the known environment (taking into account difference in Architecture)
    d. Do not focus on maximizing User Experience, just look for show stopping bottlenecks
4. Load Test
    a. Iteratively test/tune/increase load
    b. Start with about the same load as benchmark, but accurately depicting user community
    c. Do not tune until critical (primary) bottleneck cause is found – do not tune symptoms, do not tune components that "could be faster" unless you can prove they are the primary bottleneck.
    d. Re-test after each change to validate that it helped – if it didn't, change it back and try the next most likely cause of the bottleneck. Make a note of the change in case it becomes the primary bottleneck later.
    e. If bottlenecks are environment related (i.e. out of scope) document the bottleneck and present to PM for guidance. Stop testing until client/PM agree on approach.

f. ***Note*** If you need performance improved by 200% to reach the goal, tuning a method to execute in .2 seconds instead of .25 seconds will not fix the problem.

Results Reporting

1. Report test results related to stated requirements only.
    a. Show summarized data validating requirements
    b. If requirements are not met, show data as to why and what needs to be fixed/who needs to fix it by when
    c. Show areas of potential future improvement that are out of scope
    d. Show summary of tuning/settings/configurations if it adds value
    e. Be prepared to deliver formatted raw data as back-up