

When Load Tests Are Not Equal

Load testing of software applications, once a mysterious and rarified art, is now mainstream. Dozens of companies sell tools so you can bring these skills in-house, and dozens more have services that do some part – or all of it – for you. The load testing services range in *approach* from do-it-yourself to outsourced to on-site consulting, in *load delivery* from “over-the-‘net” to “behind the firewall” to “last mile”, and in *cost* from a few hundred dollars to tens of thousands.



If you’re in QA or IT and have the baton to load test a new app and are shopping for a vendor, wading through the jungle of buzzwords, hype, and deliverables to find a service that truly meets your needs and budget is a challenge.

To make sense of all the options and find the right service at the right price, like shopping for anything you know little about, you need to go back to basics: understanding – and writing down -- your objectives, requirements and constraints. Doing that right will force you to “talk to the business” in a little more detail than you may already have.

Once you’ve done that, and short-listed some options that seem to fit, then talk to the vendor and drill into the details.

Here are eight questions you should ask yourself -- and your short-listed vendors -- to make sure you are comparing apples with apples and are making a sound selection.

1. **Meeting business objectives:** Is the proposed solution customized to meet your business objectives, or “one size fits all”? Look for a vendor that asks you questions about your application and business goals that are more than “one slide deep”. A good service should be scoped and customized to your needs, and you’ll know this when the questions the vendor asks you cause you to go back for more discussion with the business stakeholders, and when the vendor’s proposal accurately portrays these objectives and answers how these will be met.
2. **Load realism:** Has the *target load* been defined in business terms, considering *concurrent sessions* **and** *completed transactions* by type? Some vendors sell you on *page views*, *real end user experience*, *geographic distribution*, *edge content delivery*, etc. – things that sound cool but may or may not be meaningful to **your** application and goals.
3. **User realism:** Will the simulated user transactions test the components of your app that real user interactions will exercise? Many self-service offerings enable you to record your own workflows, but do not enable you to modify user-input data, use unique logins, make different menu selections, etc. Without such *data parameterization*, performance results may be unrealistically fast; they certainly won’t exercise your system through to



the database, because they can be served from *cached data*. Does your vendor explain how to gauge the realism of the scripts? Will they include as one of the deliverables a *smoke test*, while you monitor your app, to verify they are loading the system and completing transactions properly?

4. **Load generation:** Where will the load come from? Will it test the app components you have control over? Some vendors will talk about driving load from many geographically distributed sites, but such results may contain internet latency times that are erratic, not controllable, and may not necessarily be material to your app/goals. Focus on whether the load driving will be from a sufficient number of unique IPs to properly load balance your web servers and provide measurements of the components of response time you can actually do something to improve.
5. **System resources:** Will you be able to find the bottlenecks? Many vendors will leave the monitoring of your system's resources to you, and fail to provide correlation of system resources with load and response times. Without correlation, you'll probably see hot spots but won't have visibility into "when" in the test and with 'which transactions' these occurred. You also risk not measuring all the necessary points to help identify the bottlenecking application/system component.
6. **Results analysis:** What's in the report? Many can take an x-ray; fewer can interpret it, provide a diagnosis, recommend targeted tuning and further testing, and set you on a course to bottleneck resolution. Ask to see an example results analysis from a project similar to yours to gauge the value it will have to your tuning process.
7. **Proposal Fluff:** Do the proposal – and contract – clearly state what you're getting? Beware the fancy proposal that contains a lot of fluff and stuff you don't understand. Look for the "meat" articulated in sensible business English. Be clear on the deliverables – from test plan to validated scripts to results analyses. Understand what resources you will need to provide to support the test. Look for milestone dates and how these match your overall deployment plan.
8. **Pricing:** Are the costs clear and focused on deliverables? Many vendors may offer less expensive solutions, or pricing based solely on *concurrent users* or *page views*. Understand the vendor's pricing structure. It should be easy to equate to:
 - The effort of developing the test and of conducting a given number of test iterations
 - The infrastructure – servers and bandwidth – required to drive the target load
 - Software costs, if these apply; web/http apps can usually be tested with free open-source tools; secure and proprietary protocols such as https. Citrix, Oracle ERP, Siebel, etc. require the purchase or rental of sophisticated commercial tools.



Assess the price in the context of questions (1) – (7). And keep in mind that effort, infrastructure and software-based pricing can be negotiated to some extent by modifying the scope of the test.

Use these eight questions to challenge yourself, your team, and your vendors, and you'll mitigate the risk of selecting a vendor that will deliver disappointment along with test results.

Load Testing Terms

Term	Definition
cached data	Data that has been fetched from the database by a previous query and is retained in memory so that a subsequent similar request can be responded to without doing a much more resource intensive disk access.
concurrent sessions / users / virtual users	Simultaneously active application sessions, each at a different step in a business process or use case, representative of how real users would appear in the system.
data parameterization	Fixed user-entered data that is replaced in the script by a variable and then driven from a data table, so that each virtual user that makes this request is different, and causes the database to do the disk access rather than serving the request from cached data. This ensures that the test simulates real users and exercises the system front-to-back (web server to DB server).
edge content delivery	Usually refers to vendor's ability to drive load from the actual locations where your users are, from dozens to hundreds of geographically distributed points of presence (POPs). This ability can be overrated, as it may or may not be important for your app or testing goal.
geographic distribution	The ability to drive load from multiple locations, simulating where your actual users access your system from. The ability to do this serves to more accurately measure <i>real user experience</i> from geographically distributed users, which may (or may not) be important for you to know.
page views [per unit of time]	The number of web page impressions that are being served per second or per minute; this is one – but not the only – measure of load of a web-based system. Two important measures that the business stakeholders usually care about -- "completed transactions" and "response time for an operation" have little or no relationship to page views.
completed transactions	Hard application metrics that business users care about and usually have good statistics about; e.g., "orders shipped", "invoices paid", "time cards posted". Quantifying these during "peak activity periods" is key to defining the target load that a good load test should strive to simulate.



real end user experience	The actual response time that an end user will “see”, which includes the network latency at the users specific location, as well as the processing time through the application. While knowing these times has some value, network latency – particularly if it is over the public internet – is subject to the dynamic routing around congested routes, and is not something you can do much about. Moreover, network latency is typically measured in milliseconds to a few seconds, whereas processing time is usually in seconds to tens of seconds. Optimizing the processing time, which you can diagnose and tune, usually has more business value.
smoke test	In the context of a load test, a smoke test is designed to “test the test” to ensure that scripts are truly completing transactions and exercising the system as real users would. Without a smoke test monitored carefully at the application end, scripts may be failing to generate the expected load, sometimes even without returning an error that would be evident to the tester.
target load	The load that a test should attempt to simulate, comprised of all the key metrics relevant to the application, usually including at least <i>concurrent sessions</i> and <i>completed transactions</i> by type during a peak processing period.

ABOUT Mentora

Mentora tests, hosts and manages business applications. We provide software quality and production management solutions across the application change lifecycle. We specialize in performance testing of enterprise apps from Oracle, SAP, Trizetto/Facets and SunGard EXP. We host and manage business-critical applications in eCommerce, Healthcare, Insurance and Publishing at top-tier AT&T data centers. Headquartered in Atlanta, Mentora has offices in Boston and Washington DC, and delivers services nationwide.