

Best and Worst Practices?

Ask the Oracles!



Gaja Krishna Vaidyanatha:

Wikipedia defines best practice as “a management idea which asserts that there is a technique, method, process, activity, incentive or reward that is more effective at delivering a particular out-

come than any other technique, method, process, etc. The idea is that with proper processes, checks, and testing, a desired outcome can be delivered with fewer problems and unforeseen complications. Best practices can also be defined as the most efficient (least amount of effort) and effective (best results) way of accomplishing a task, based on repeatable procedures that have proven themselves over time for large numbers of people.”

Those of you who have been there and done that with Oracle systems for many years know very well that it is impossible to predict and plan for every scenario that your application and database will present in production. This is especially true when you embark into upgrading any component of your system—especially when you are betting on a new feature in the software to bail your system out of a sticky situation! But you upgrade anyway, because inaction is worse than the current performance status-quo or corner-case bugs that you may encounter in the future. Best practices come in very handy in situations such as these.

In a recent performance management engagement at one of my customer sites, I was privy to a strange (yet seemingly common) situation. The Oracle database was upgraded from 9i to 10g, and even though all of the pre-upgrade load tests performed in stellar fashion, the database experienced sporadic hiccups in performance during the post-upgrade phase in production. And during these hiccups, the CPUs on the system were constantly and consistently maxed out. This was because the execution plans of the core SQL in the application had gone south. When a 32-CPU machine with an average idle percentage of 85% in Oracle 9i gets maxed out in Oracle 10g, you know there is a gremlin lurking out there. With adequate performance diagnostic data, it was determined that Oracle’s “automatic feature” of “peeking for bind variable values” at hard-parse time was the cause for the pain. To ensure future stability and to guarantee performance capacity during the 2007 holidays, the feature was completely turned off.

How do you plan for a situation such as this, when it did not surface during pre-upgrade performance load tests? Clearly, in this case, the pain had to be experienced before remedial measures could be undertaken. But does that mean that you don’t test? No, you still do. Nor does this mean that you take preemptive action and turn off every new feature or modify the default values of every Oracle initialization pa-

rameter that is different from the prior release—*especially* when it is an “underscore” parameter.

From an Oracle Performance Management perspective, the following are some key best practices I’d recommend:

1. **System Performance Diagnostic Best Practice**—Chant and practice the “response time mantra.” At the end of the day, performance management is about user experience and application response times, not a slew of database health metrics. If you engage with a specific response time goal in mind, you will stop tuning when you achieve that goal. Remember, Compulsive Tuning Disorder (CTD) is not a good thing!
2. **System Performance Diagnostic Best Practice**—Utilize mathematical data instead of expert opinions to arrive at *all* of your conclusions. Opinions are not facts and can be very easily protected by the expert with the YMMV (Your Mileage May Vary) camouflage. When you can categorically state that a query’s elapsed time is 15 minutes, and 83.5% of its elapsed time is spent in performing single-block I/O requests, then that is a quantified problem that you can attempt to solve.
3. **Application Performance Testing Best Practice**—Create a realistic test environment, so that major upgrades can be tested for feasibility, reasonability, and stability. Production life is hard enough with testing, and you truly do not want to be out there without testing. Your load tests should reflect reasonability and veracity when compared to your production environment. Testing should be done using realistic production data (or at least production Optimizer statistics), from both the database and application’s perspective.
4. **System Performance Environment Best Practice**—Engage in balanced maneuvers when building and upgrading systems. If you double the capacity of your CPUs, then take the time to review the capacity in your I/O sub-system to determine whether it too requires a similar upgrade. Throwing more or faster CPUs without reviewing the capacity of your I/O sub-system may result in real-time system imbalances. These imbalances will translate into long-term scalability and performance problems.
5. **Application Performance Design Best Practice**—If a piece of application functionality is data intensive, build it with SQL first. If the code requires complex logic, then build it in the database (yes PL/SQL). Alternatively, if the functionality is processing intensive, then build it in the application tier. In either case, avoid unnecessary single-row processing roundtrips between the database and the application tier.

Based on my experience in the field, I'd recommend that you steer away from the following worst practices:

1. **Building “database agnostic applications”**—This is a commonly used buzzword line that does not make any sense. The term “agnostic” means “to neither believe nor disbelieve” and to limit one’s belief to human experience. Are we saying that if we reinvent the wheel and build database constraints in the application tier, we are actually limiting our belief in Oracle based on our experience? Nonsense! Databases such as Oracle should not act only as “data stores,” as they provide way more functionality for scalable performance. Not utilizing the inherent power of the RDBMS is like driving a Lamborghini with 95% of its power turned off.
2. **Overusing dynamic SQL at the application tier**—If Oracle has to parse every single SQL statement that you present to it because of the dynamism that you have introduced in your application logic, it makes your system that much less scalable. There are application environments where CURSOR_SHARING cannot be set to FORCE and dynamic SQL (non-reusable SQL) will open up a slew of library cache and shared pool latch contention performance problems.
3. **Designing without scalability**—If you have the luxury of custom designing any part of your system, not ensuring that it will work with large data sets is a worst practice. For example: if you write a query that works well with a one-thousand-row table, not ensuring that it will exhibit a reasonable response time when processing a one-million-row table is a recipe for disaster. The underlying theme here is “design for scalability.”
4. **Using database ratios to determine the true performance bottleneck**—Ratios may be indicators but definitely not “root causes” of performance problems. In all of my engagements with my customers, I have yet to find the need to review a single database ratio to solve a customer’s performance problem. The 10046 trace files combined with STATSPACK and AWR reports and some OS commands pretty much define my tuning arsenal. Yes, I do skip the ratios in the aforementioned performance reports. And if I do find ratios to be useful in the future, you will be the first to know!
5. **Drinking vendor Kool-Aid due to their stature in the marketplace**—Verify the durability and performance of any piece of software or hardware by putting it through the paces in your test environment. Again, remember Best Practice #2—Use mathematical data, not expert opinions. And don’t fall prey to sales pitches!

So in the context of Oracle Performance Management, adhering to the best practices listed in this section is a good thing. And by no means is this a comprehensive list of best and worst practices; it is just a beginning. May I remind my readers: “Well begun is half done!” Cheers! ▲

Gaja Krishna Vaidyanatha has over 15 years of industry technical expertise working with Oracle systems. He is the principal of DBPerfMan LLC (www.dbperfman.com), an independent consulting firm specializing in the area of Oracle Database Performance Diagnostics & Management for

Fortune 500 corporations. He is the primary author of Oracle Performance Tuning 101, published by Oracle Press, and one of the co-authors of Oracle Insights: Tales of the Oak Table, published by Apress.