Essential Roadmap to Benchmarks for DBAs

Essential truths that parallel a benchmark:

- Everyone is somewhere.
- They will end up somewhere, whether as leaves in the wind, as nomads in the desert, or the navigator of a ship.

This is a roadmap to surviving a benchmark and accomplishing your goal with less effort. Simple mistakes can change the project duration from weeks to months.

How did you get here?

- Your current Database System is “out of gas” and can no longer meet business expectations
- Your current Database System is being discontinued
- A new CIO or Architect is convinced that “cloud” is the answer
- A software or hardware upgrade demands testing before going “Live”

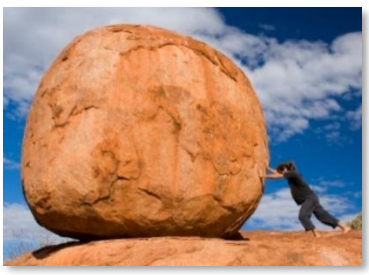
How will the final decision be made?

- If this is just a software or hardware upgrade, the decision can be made by the DBA or a small project team. This is the simplest use of a benchmark.
- If the benchmark will result in a major change in technology,
    - executives will need to be sold on spending on this technology versus other business investments,
    - organizations using the existing platform will need to be sold on the conversion, training and potential delay of other business intelligence development
    - technologist will need to be sold on the chosen new technology versus ones they may be more familiar/comfortable with.

It is critical to begin with the end in mind and those final steps leading up to the decision. For the major change in technology, you will need to

- identify business benefits of the change,
- show how the existing workload will be better served by the new technology
- show what new capabilities will be facilitated by the new technology.

You can’t just select 10 queries and see which platform is faster as input to the final decision



As an agent of change, you can apply energy to overcome organizational inertia before, during or after the benchmark. If attempted before the benchmark, you need an impossible level of credibility across the organization to believe you. If you wait until you have analyzed the benchmark results, people will poke holes in your analysis. The decision makers and victims of the change (users) need to be involved in defining critical success factors and participate during evaluation “to see for themselves”.

Benchmark Methodology In 9 Steps

1. Look at the ceiling
2. Acquaint yourself to today
3. Argue & agree on **C**ritical **S**uccess **F**actors (CSFs)
4. Gather tables and queries
5. Get the data
6. Adapt queries for benchmark
7. Rehearse the benchmark
8. Execute and measure
9. Analyze & present

1. Look at The Ceiling

This is another way of saying to visualize a future state with its attributes that addresses your current pain points and missed opportunities using user current technology. As your vision of the future crystalizes, it is important to start a list of critical success factors. Likely you have been following various publications about data warehousing, perhaps attended conferences, and discussed ideas with associates and users. Capture those critical success factors and define specific measurements that would unambiguously drive your decisions.

A benchmark without measurements is a decision on a foundation of biases, influence and power.

2. Acquaint Yourself to Today

There is a current state and a future state. As you are finishing up on #1 you'll have an idea of the target future state. You need to be well grounded in the current state. It is surprising in years of conducting benchmarks, how often the people leading the benchmark can't answer basic questions about the existing workload like:

- How many total users? How many users by Organization?
- In the past several months, when were the peak workloads and how many users were connected at those times? How many queries were actually in flight?
- What is the response time needs of different groups of users? (Can they even distinguish finance users from sales users from marketing users? Reporting users from data scientists?)
- How often are those needs not being met? What is the impact to the business?
- How much workload and data has shifted to other platforms because of limitations of the current platform? How much workload is just exporting to other platforms?
- While the one question they can usually answer is how much data they have, often they can't answer the question about projected data size in one year.

Why is this important? The current production workload demand will need continue to operate on the new platform. The platform that can execute 10 queries faster than another may not support the concurrency, or the mix of workload priorities needed now and in the future. Focusing on performance of 10 or even 100 selected heavy reporting queries ignores thousands of trivial BI lookup queries that can challenge some DBMS's.

3. Argue & Agree on Success Criteria

This is a point where you should consider if this is a "Pay me now or pay me later" situation.

- If the results will drive the decisions of one person in an area where they have authority, such as a DBA deciding on whether system settings will help system performance, then there is no inertia to overcome
- If the results will cost a large sum of money to acquire, convert applications, and train support personnel and end users, then there is lots of inertia to overcome. Beyond the measurable costs, there is comfort, pride, fear and a few other emotions contributing to the inertia plus the fact that everyone except the team doing the benchmark have a primary day job and they will be distracted from their primary mission to participate in the change.

In the latter case above, the time spent getting the community to agree:

> "if the technology can do a, b, c then I would support the change and set aside comfort, pride, fear, etc."

then that is time you won't need to spend at the end selling them on the benchmark results and having to overcome suspicion and doubt.

In competitive benchmarks, if the customer or prospect can do #1 (visualize a future state) and #2 (profile existing workload and SLAs), we can facilitate a workshop to assist in #3. This doesn't have anything to do with the technology but rather getting clarity on the objectives. With the measurable critical success factors laid out from #1 and an idea of the current environment, we can help define a list of tests that support those critical success factors.

4. Gather Tables and Queries

The best workload for a benchmark is derived from actual production workload. The mix of selected queries should be representative of production based on number and size of tables, complexity of joins and functions, execution time and resource consumption. Data warehouses aren't read-only. Some ETL/ELT activities should be included.

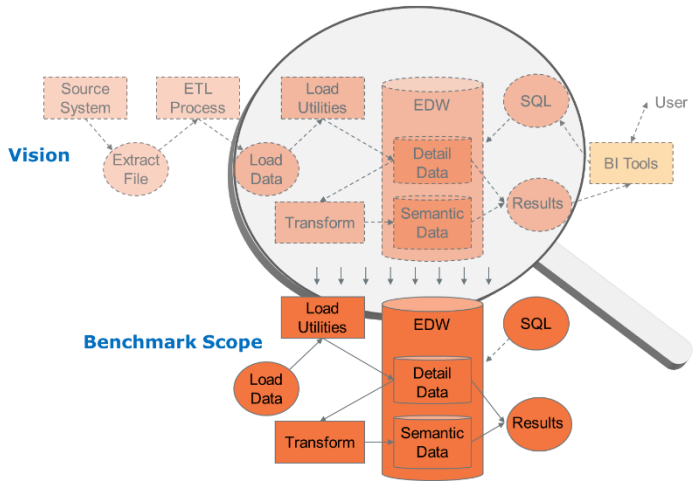
When we are asked to assist in the selection of the queries for a benchmark, we:

- Ask the customer/prospect to identify some key tables or workloads
- Analyze the hourly workload over a month to identify key periods for sampling queries
- Analyze the queries in flight every 30 seconds in those periods to assess concurrency needs
- When BI tools are in use, a report is constructed using multiple queries with intermediate temporary tables, so those must be selected by session. Other types of queries may be usable as stand-alone queries.
- Analyze the other tables those sessions/queries would need
- Analyze the number of queries accessing each table and eliminate sessions that require the addition of large tables not needed by other sessions
- Validate the selected session selection based on number of tables referenced, query steps, use of functions, processing duration and resource consumption as being representative of real production.

We also ask for a few ETL/ELT jobs. To be realistic, if they normally update data at the same time users are accessing it, then they should execute that way in the benchmark.

You should avoid "boil the ocean" with far more tests and effort that is sufficient for making a good platform decision. Often, significantly more tables are exported than are used. Using database object logging can help identify data actually needed, reducing export and loading effort. In one case we reduced 80% of the data exporting and loading effort. (Don't forget that exporting data can be extra workload on a production machine)

Often customers want to run a benchmark using their BI or ETL tools. The problem is that those tools run on a different platform from the DBMS and have network connections. Those other platforms may have uncontrolled competing workload and network congestion. If the benchmark is to choose between alternate DBMS's, not only is it important to have a repeatable workload across vendors, but it is important to be able to focus on the DBMS's contribution to performance without hiding it in some other performance issues.



In one competitive benchmark, we were forced to work with a POS vendor's application running on another platform. At the customer visit to the benchmark center, on one screen I had a monitor displaying DBMS activity and during the half hour test, there were a few blips of DMBS activity … probably not more than 5 minutes total. Comparing the tests between DBMS vendors, one could conclude that the DBMS choice didn't matter since performance was limited by the client application.

If you are trying to choose between DBMS's, focus the benchmark on the DBMS and not everything else. While there are several approaches to simulating the workload, using a tool like TdBench that works across DBMS platforms and is designed specifically for simulating benchmark workloads is a good choice. It supports multiple work queues, parameterized and prepared queries, paced and scheduled query execution, integration with OS commands and load utilities, built-in reporting and linkage to host DBMS query logging. It can run on Windows, a Linux server or even a database node.

5. Get the Data

If queries are run against live data in a production system, it is difficult to get valid comparisons of performance if the data being queried isn't the same. For the DBA validating an upgrade, running the test several hours apart should not be a big deal. If the tests are run days/weeks/months apart, not only is the data being processed different, but users and BI queries generate SQL with specific date/time references that may no longer valid next week or next month.

It may be necessary to take a snapshot of the data into databases dedicated to the benchmark. Rather than dedicating large amounts of space for the benchmark, in the past, we've taken a snapshot of the smaller tables, a sample of the larger tables and built scripts to build a target benchmark database at the

scale needed for a given benchmark execution: small for a development platform and large for validation of a new production platform.

If the benchmark will be executed by people that do not normally have access to the data in scope, steps should be taken to protect confidential and PII data. You may not be able to simply wipe out those columns containing identifiers if those identifiers are used in the constraints and joins of the queries. You will need to map join columns to meaningless values to maintain referential integrity. Sensitive textual columns (e.g. names, addresses) can be scrambled (e.g. randomly reassigning fist name, last names, address numbers and streets to different rows), however queries that use constraints against those columns may need to be modified to find a meaningful result set of similar size.

If you are taking data out of the production platform to move to a different platform that is not under production controls, you need to get approval. Even if you plan to scrub confidential and PII information, the data owner, operations and corporate information security may need to approve the plan and there may be a need to audit the results of the data scrubbing. If data is being exported to a transportable media (USB, NFS, etc), you may need to get your plans to encrypt the data approved as well. **This can take weeks if properly executed and months if approval steps are overlooked**.

For a competitive benchmark, the snapshot will be in the form of a delimited data export. Huge amounts of time are wasted if the export is not well designed. Key criteria:

- The delimiter must never appear in the text meaning in 98% of the cases, “comma delimited” using real comma doesn’t pass this criterion. “Pipe delimited” a.k.a. vertical bars … | … are often used but occasionally in call center comments or product descriptions, those may appear. Tilde (~) is another option. If in doubt, choose a character nearly impossible for someone to enter from the keyboard, like the beta character (β). Using commas in a recent export caused a re-export of 75 TB in a recent benchmark and 3 week delay.
- A record in the export file will be loaded to become a row in a database. The problem is that text from call center and web applications may contain line end characters resulting in an incomplete row. Enclosing those in quotes … “ “ … is a common protection, but then the text may contain quotes causing problems for the load utility. Some utilities support an escaping character such as the back slash … \ … such that \” will be taken as “ and \\ will be taken as \. Some utilities will allow imbedded quotes to be doubled such that “abc””def” will be taken as “abc”def”, but that poses problems with empty text fields that are properly exported as “”. Failure here will either require re-export of data or complex, slow processing to glue rows back together that will only be partially successful.

- Dates, Times, and Timestamps should use standard, fixed format representations so 01/01/2018 is preferred over 1/1/18, 07:15:05 is preferred over 7:15:05 and 2018-01-01 07:15:05 would be a preferred timestamp format. Some databases drop the seconds if they are :00 which can pose problems on import. Failure here will cause you to import data as text and create data transformations to standard date forms which costs valuable time.

- Numbers should be represented with their intended precision. 2.10 is preferred over 2.099999999999 from a floating-point field.

- In a proper relational model, nulls and a zero-length string are not the same thing. If nulls are properly used in the source database, then nulls need to have an explicit representation in the export. Often, we see used the text string: null

6. Adapt Queries for Benchmark

Queries will need to be adapted to work in a benchmark test. If you don't, SELECT statements may return zero rows if they look for shipments processed today in a snapshot of data taken 2 months ago. Testing Update queries on a production platform could destroy production data integrity.

- References to tables will need to point to the databases being used for the benchmark
- References to CURRENT_DATE, CURRENT_TIME, and CURRENT_TIMESTAMP will need to be changed to constants that are consistent with the time-period of the data snapshot.
- Multi-query scripts that create work tables in a database should be converted to use temporary or volatile tables such that if multiple copies of the same script run at the same time in the benchmark, they won't interfere with each other.
- To support analysis of which queries run longer, queries need to be given an identifier. Putting a comment string after the verb with an identifier works across any DBMS. For example:

  ```
  Select /* tdb=qry021.02 */ inv_dt, inv_num, ship_from, …
  ```

  Or

  ```
  Select /* tdb=:queryname */ inv_dt, inv_num, ship_from, …
  ```

The marker :queryname is used by TdBench to use the script file name as the identifier.

If you have a modest number of query variations, you can parameterize the queries and extract a set of data values that when substituted into the query template will produce rows. For example, while you could get a list of valid customer numbers from the customer table, if the queries join those customers to the order table, your query to build the parameter file should do a part of the joining logic from the query to ensure that the selected customer will meet all other criteria to produce rows.

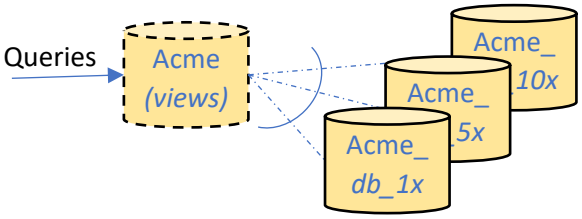
For ETL/ELT scripts, you must have a way to reset the tables for each test. These always must use databases and tables dedicated to the benchmark. If ETL/ELT happened against the same tables being queried in a production environment, then the queries should go against the databases and tables dedicated to the benchmark. To reset the tables, between each test, your options:

1. Delete the modified tables and copy from backup. This is best.
2. If data is being appended to the table for a more recent period, delete the date range or drop the partition. If the new data at the physical level is being merged into data blocks, then the first time will require block splits, the deletes will leave empty space and the next test may run faster since block splits aren't needed. Some databases don't actually remove deleted rows so as testing proceeds, the target tables become larger and larger.
3. In some cases, a column is used to mark changed rows so the changes can be reversed or deleted.

Net: The easiest for repeatability is option 1.

You also need to collect the transactional data that will be needed for the ETL/ELT scripts. This data may disappear from production before you can get it, so the timing of the capture of that data needs to be coordinated with the timing for taking the snapshot of the target tables.

If the benchmark will be running against different scales of data or tuning options, our best practice is to change the tables to views and have the alternate tables in different databases. That allows a simple script to change the views in the target database to point to a different version of some or all tables.



Finally, make sure you've got DBMS rights granted between view database, table databases, and the logon ID's that will be running the streams of workload.

7. Rehearse the Benchmark

There are a series of tests that should be executed before you get to the formal execution of the benchmark.

Test 0 – If you are constructing databases and tables dedicated to the benchmark, the first test should be against empty tables. That will answer in seconds if you have assembled all tables, views, and access rights needed to successfully execute the gathered workload. If you need to load data, this run against empty tables can provide data from query logging about what tables are being referenced to reduce the effort to export and load data. If you are adapting a data model from a different platform, this provides an opportunity to determine the pattern of column usage for constraints and joins to adapt the tables before loading.

Test 0.9 – After you've loaded the data and before any extrapolation of the data, a serial of all queries in a single stream will help to identify performance outliers, ones that fail and ones that return no rows. Both failures and zero rows returned happen in a production environment. Errors are generally pointless to leave without fixing them or removing from the benchmark. Zero row results should not happen at a rate different than production environment you are trying to simulate.

Test 0.9.0 – After any extrapolation and data is complete with collected statistics, the serial test will tell you the best the platform can do if there is only one user on the system. This test shows how the DBMS can break down the workload into parallel activities. The percentage utilization of CPU and I/O on the serial test will tell you the maximum number of query streams that can run concurrently (theoretically) before the DBMS needs to be allocating CPU and I/O resources to more demand than it has available.

Example: If the average CPU utilization is 10% for the serial test, then at 10 query streams, you would expect (theoretically) 100% utilization and if you double to 20 streams of queries, each stream will get ½ the resources it got previously and take 2x as long. However, ideally you should get the same number of queries executed per hour. In this simple situation, a good workload test suite would be 5, 10, 20, 40 and 80 stream tests (representing 50, 100, 200, 400 and 800 logged on users) and with each doubling of the number of streams, there should only be a moderate decrease in the number of queries completed

per hour. But if you have 3200 users, should you run 320 streams? No unless the purpose is seeing if the database will crash. Unless the queries use a single unit of parallel processing, you won't increase the QPH after you reach the limit of CPU or I/O. The response time will merely double as you double the streams.

The longest running most resource intensive queries will also tell you how long the workload tests will need to run. If that query needed 50% of the resources on average during its 10-minute run, at 10 streams, with it only getting 10% of the resources, it will run 50 minutes, suggesting either you run the workload tests for 60 minutes, accepting that the query may not complete at 20 streams. Another option is to only test the really heavy queries in the serial test.

Test 0.9.1 – Try out a few of your workload scripts and analyze the results. In some cases, you'll find that queries collide on the same resource due to locking. Look at completions and partial query executions at the end of your test to determine if you've picked the right duration for the test. If you got a huge number of executions and all queries complete at least once, consider shortening the duration of the workload tests. You can be much more productive with 5 or 15 minute workload tests than having to run 1 to 3 hour tests. You can always calculate **Q**ueries **P**er **H**our (QPH) by using the ratio of your test interval to 1 hour.

8. Execute & Measure

The formal measured tests will ideally occur on an idle system with no competing workload. If you've extracted the queries from the BI and ETL tool and are running all scripts under TdBench 8.0 on a client with low latency to the DBMS, you've made a huge step at getting repeatable, reliable results.

Continuing this idealistic journey … Databases naturally cache database blocks and some even retain partial results in memory. Some customers/prospects have requested that the database be restarted for each test. If a feature of the database is to improve performance thru caching, then simulating a "realistic environment" by forcing the database to perform unnatural I/O doesn't make sense. It is better to address the caching issue with a robust set of queries and/or use parameterized queries to ensure all data gets touched.

For repeatability, you could start out a test series with a serial test and follow with the workload ramp-up tests. For the tests with ETL, building the reset of the target tables into the TdBench scripts ahead of the DEFINE statement will ensure those tables are always at the same state for each test.

If you don't have an idle system or an environment with other potential network and client interferences, you need to:

1. Run the test series at different times of the day to see if the results are repeatable. On one benchmark, we got dramatically different data loading performance on a weekend night and didn't realize the client server and the DBMS were connected via a network that was severely overloaded. We should have been suspicious because the full-duplex echoing from Linux as we typed had occasional hesitations.
2. Use workload management to dedicate a percentage of the resources to the benchmark when it runs.
3. In every analysis, include the non-benchmark workload and use case statements to label it as "Noise" so you are always aware of things that may be impacting your results.

It is important to do a quick validation of the tests as you execute them to make sure that the results you are gathering will be useful for your analysis and presentation. This is especially critical if you can't keep each of the environments available throughout the benchmark, either because of space limitations on one platform or because for each part of the benchmark you'll be reconfiguring the system to add more nodes.

If you've done a thorough job with the steps above, you shouldn't need to make changes to the test suite as you execute the tests against each platform. It is unlikely that anyone can build a benchmark environment in a few weeks and run tests without having mistakes pop up. It is not unusual to have hundreds of RunID's in the TdBench TestTracking table with only 20-30 of interest to the final analysis and presentation. It is highly recommended to use the NOTE command to document observations about tests, especially when they fail so you can be productive when you get to the Analysis. Better yet, as you get RunID's that you believe to be final based on your quick validation test, use the NOTE command to append a string to the notes you can use to filter those out using a query against the H2 database.

9. Analyze & Present

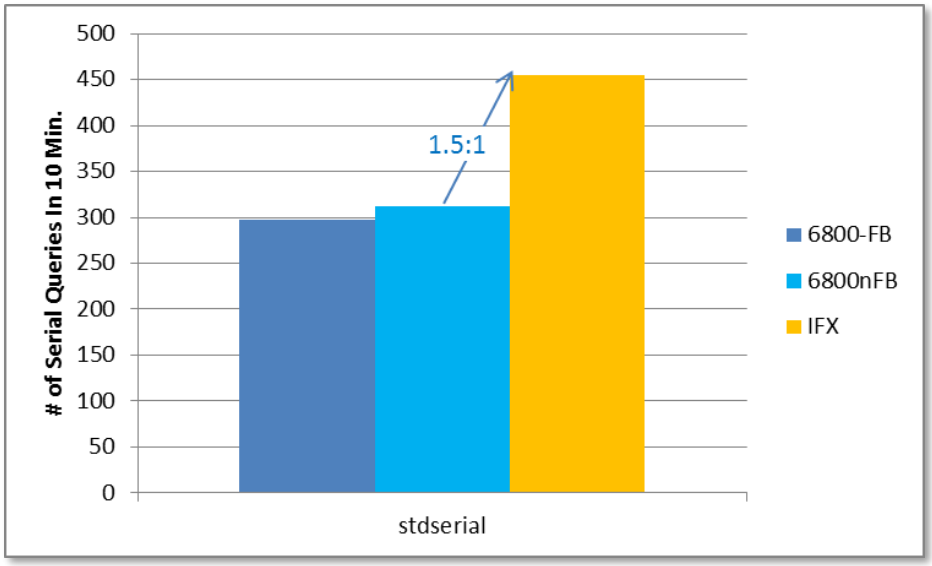
If there was a thorough definition and agreement to the measurable critical success factors and you've marked the good tests with NOTE command, the only thing that remains is determining how to distill that information down to the level of detail that your audience can understand.

- On your side, being immersed in the details of the execution, you have so, so much to share.
- On the other hand, Executives may be looking for a single slide.

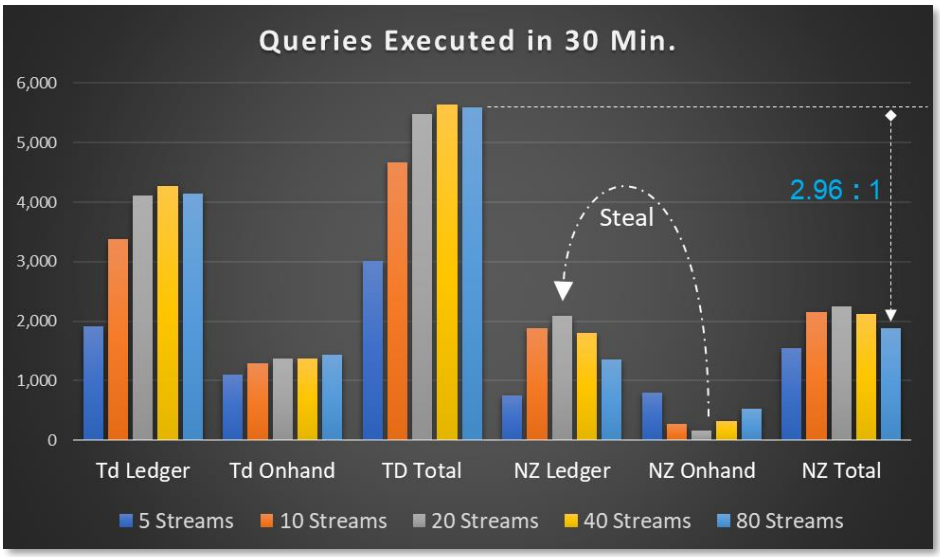
The objective of the analysis and presentation is to distill the mountains of collected data to prove how the subject of the benchmark meets the critical success factors identified in Step 1.

It is beyond the scope of this user guide to instruct you in how to analyze and present results. There are a few pointers to keep in mind.

- The context of numbers is critical to others understanding results. "35 seconds" may be great for a data mining query and awful for a Call Center retrieval of customer balance. It makes it much easier to understand if the timings are compared against the existing system. While that system may not be an option for the future, it can provide essential context.

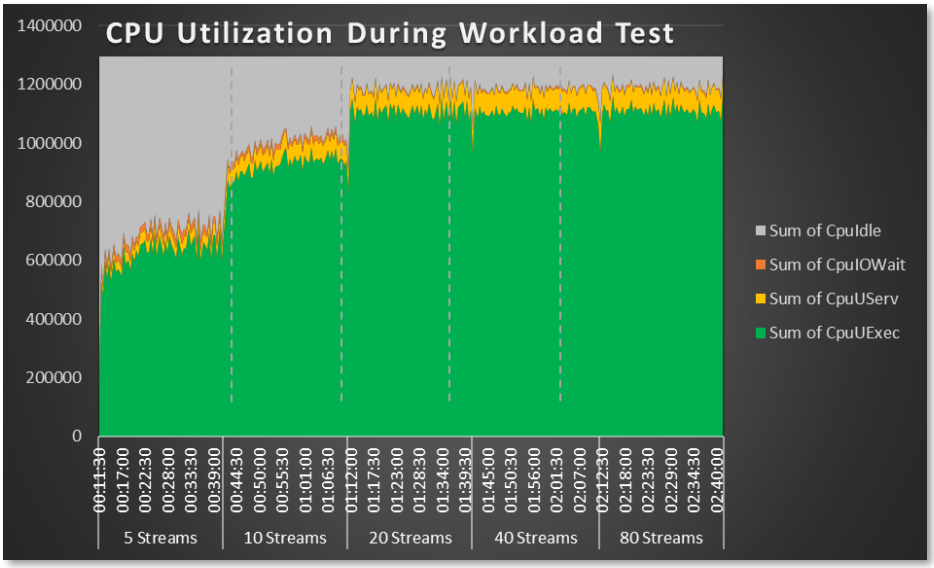


- It helps to group queries under a name which has meaning to the business.
- The primary function of a benchmark gives context to numbers. Make it easy for audience to compare queries per hour at 5, 10, 20, 40, 80 streams for each DBMS or platform size.

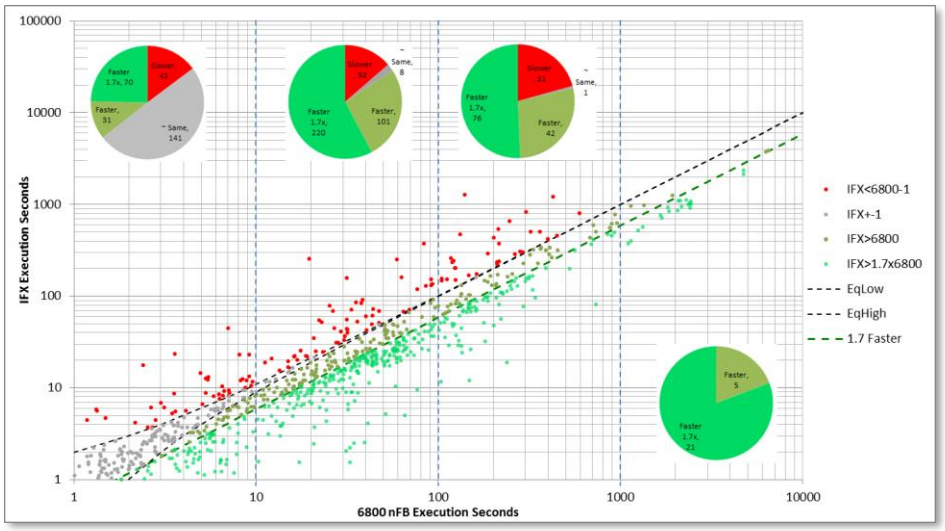


(More queries better)

- Limit slides to 10 rows of numbers per slide. (The Microsoft engineer that thought 8 point font in PowerPoint was a good idea should be shot)
- Focus on numbers meaningful to the audience. Response time and queries per hour have meaning. CPU seconds and I/O count probably don't unless expressed as a % of total capacity.



- If you have hundreds of queries to compare on performance between 2 platforms, use a scatter plot with response time of one on the X axis and the other on the Y axis. Draw lines bounding regions of "n%" better or worse and add a comment with number in each region.



- Begin the presentation of results with the list of critical success factor. End the presentation with the same list with annotation of how you met each one

Concluding Remarks

If you've worked the process to bring user groups and decision makers along as you planned, designed and executed the benchmark, the decision should be easy, and you'll have selected a platform that will meet your company's needs in the coming years.

If you haven't been able to engage them during the process, expect additional weeks/months of effort to get people on board with your selection.

If you used 10 queries and chose the platform that could execute those queries fastest instead of modeling the real workload demands, recognize that you probably wasted time and money doing the benchmark. Just because a football linebacker is tops at making touchdowns, if faced with having to do a 500 meter freestyle race on the swim team, don't expect success in one activity to translate into success in a much different activity. He might not even know how to swim. The platform winning the 10 query benchmark may drown when faced with your real and planned workload, and you (or your replacement) will be back selecting a DBMS in months.