ARE WE DONE YET?

CERTIFYING APPLICATION PERFORMANCE UNDER NEW DATABASE VERSIONS OR ARCHITECTURES

David A Haas
The Nielsen Company
Session #421

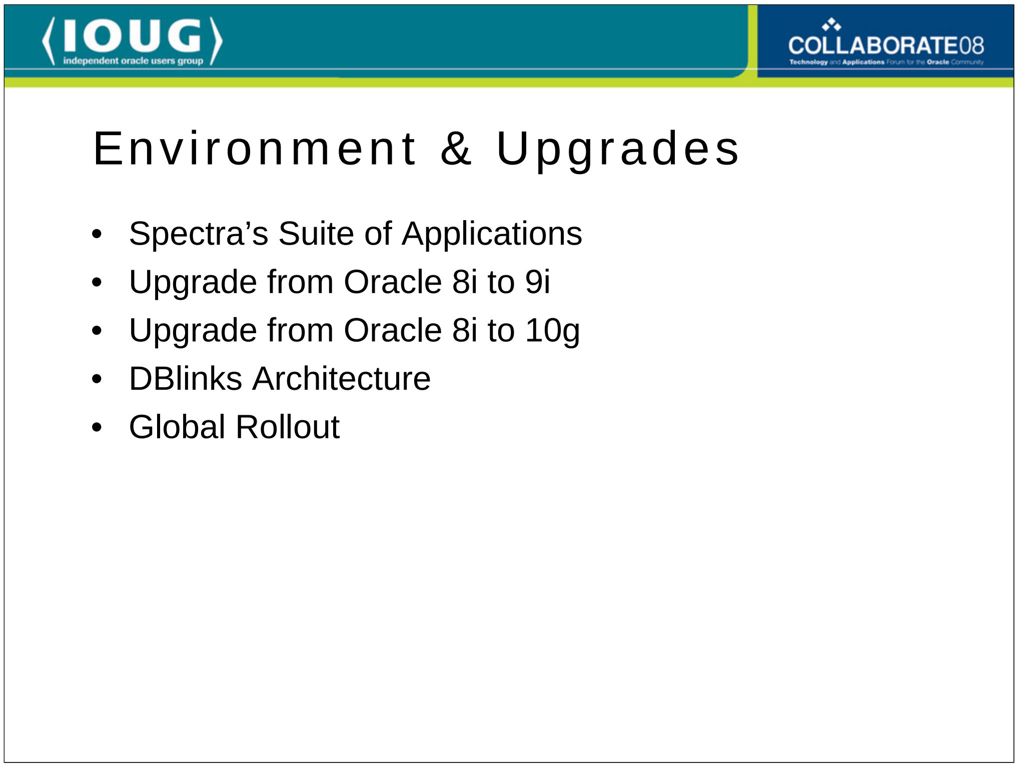
Speaker Qualifications

- Senior Director of Database Architecture serving the Consumer Panel Services group within the Nielsen Company.
- With over 20 years of experience with Oracle relational database technology, I have participated in the construction of data collection, automated data production and decision support systems in a number of roles during the growth of an entrepreneurial startup company into a business unit within a global provider of marketing information.

nielsen

Overview

- Fear can cripple a cautious development team and prevent upgrades necessary to maintain business and technological relevance
- Lack of knowledge can allow a reckless development team to make upgrades that render their application unusable
- Application Development teams require proven techniques to evaluate the impact of a change
- Empowering them to make the calculated risks necessary for continued relevance



Environment & Upgrades

- Spectra’s Suite of Applications
- Upgrade from Oracle 8i to 9i
- Upgrade from Oracle 8i to 10g
- DBlinks Architecture
- Global Rollout

Spectra’s Suite of Applications

- A web based marketing analytics system; combining proprietary methodology with in-house and third party data staged in an operational data store, which generates individual data marts focused by user selections
- We typically refer to the data marts created by our application as reports, however, they contain data which is not directly available in the database, created based on parameters supplied by the end user targeted to resolve a specific sales or marketing issue.

Oracle 9i & 10g upgrades

- The main goal was to maintain support from Oracle combined with underlying partition structure eliminating some weaknesses within the current range partitions along with increased block size for the fact tables within the schema

DBlinks Upgrade

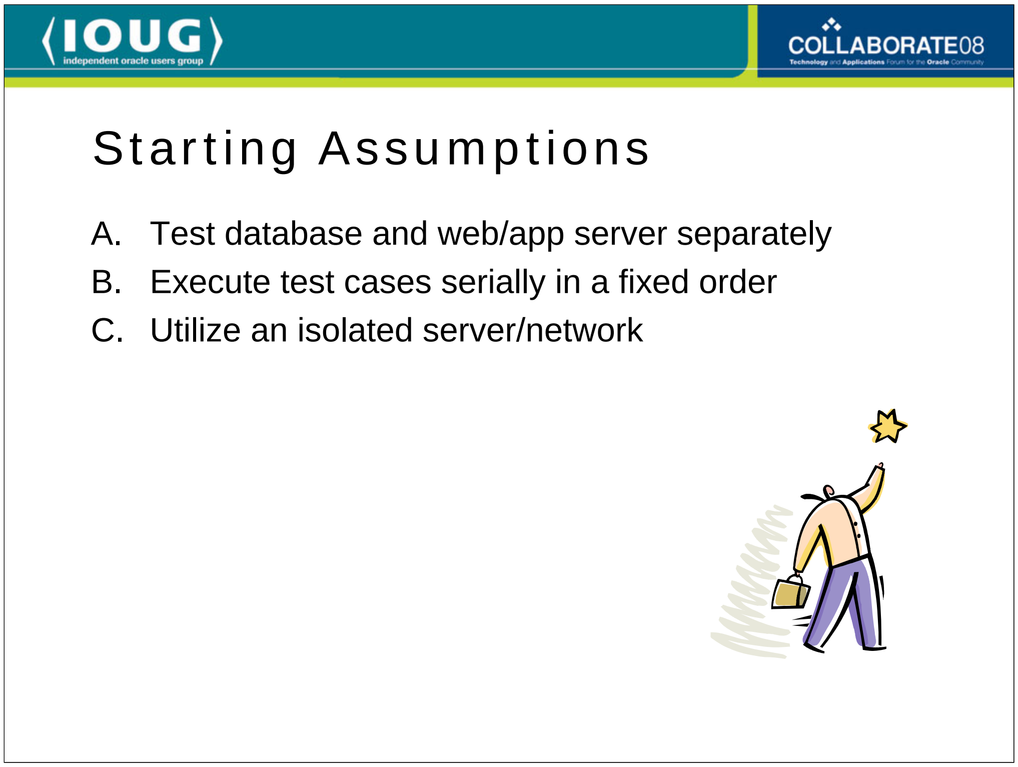
- Radical shift in database architecture isolating static and dynamic data content into separate instances presented to the application as a single virtual instance using remote database links
- Eliminating the largest part of the data release process, synchronization of user created assets. We are now able to release updated data in minutes at any time during the week. Prior to this, we were limited to releasing data on the weekend with 12-18 hours of downtime, depending on users not creating new assets for 18 hours prior to the update.

Global Rollout

- Our main concern in the upgrades was maintaining current performance levels, while simplifying our support processes, increasing the quality of the deliverables and eliminating end user downtime in preparation for a global rollout of our solutions.

Outline

1. Starting Assumptions
2. Test Case Selection
3. Benchmark Process
4. Proposed Environment
5. Final Comparison



Starting Assumptions

A. Test database and web/app server separately
B. Execute test cases serially in a fixed order
C. Utilize an isolated server/network

Test Components Separately

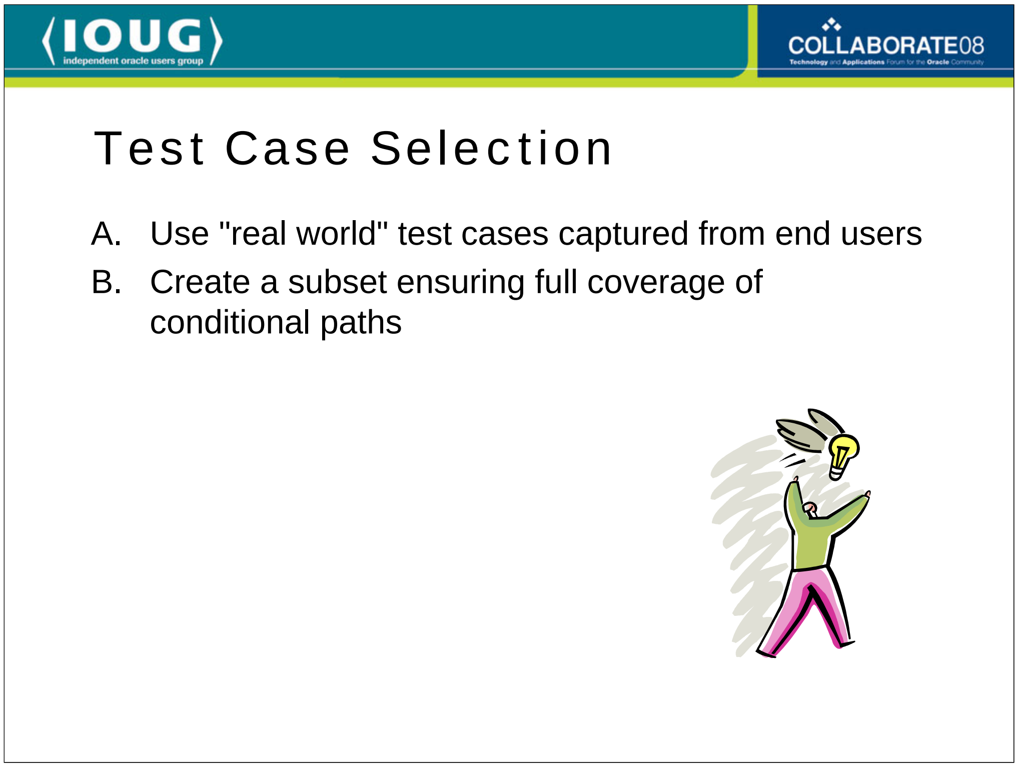
- Ultimately end user perception of the system is all that matters
- Impossible to capture performance benchmarks from this point of view
- Focus on parts of the system impacted by change
- In our case; running Java method calls directly on database server

Execute Test Cases Serially

- Compute statistically reliable estimate of average performance for each test
- Execute in consistent order and serial fashion to eliminate caching differences
- Tests running in parallel impact the environment for every other test
- Running individual cases serially reduces external performance impacts

Utilize an isolated Environment

- Removes variance from network and application server layers
- Ability to isolate database depends on storage environment
- Accept variances for remaining tests that do not converge towards statistically reliable estimate of average performance



Test Case Selection

A. Use "real world" test cases captured from end users

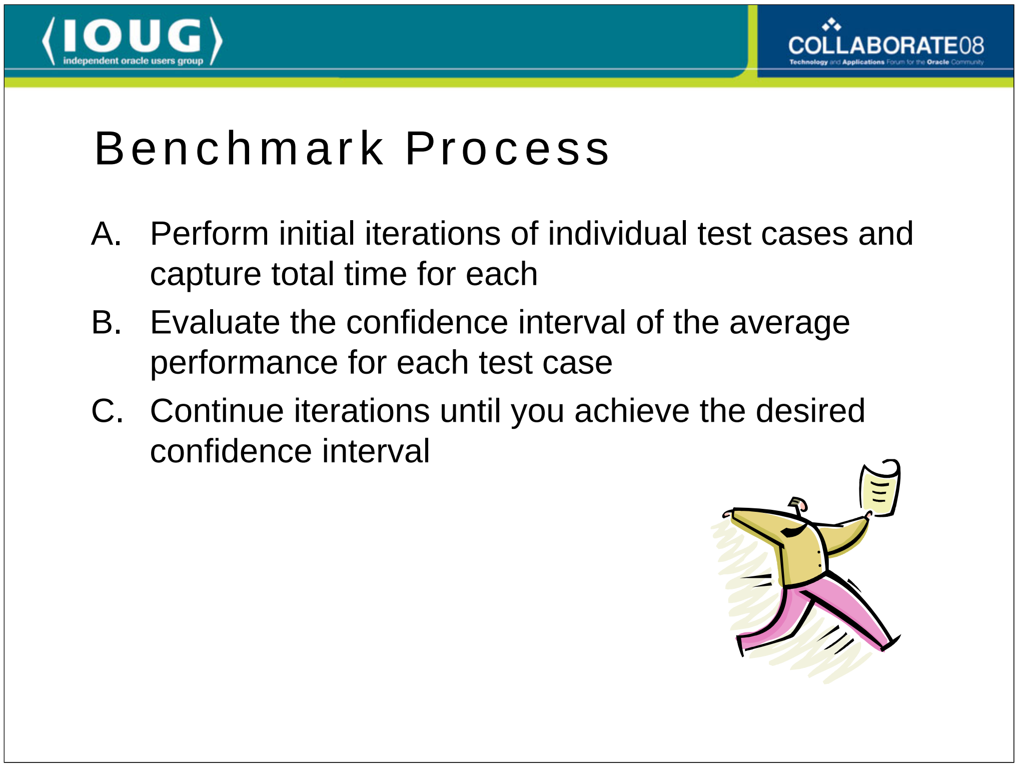
B. Create a subset ensuring full coverage of conditional paths

Use "real world" test cases

- ➢ Logging tables within our application capture all end user report requests
- ➢ Extract report requests submitted by actual client users, not internal folks
- ➢ Limited to reports from the past three months, ensures data availability
- ➢ Covering a cross section of client usage patterns
- ➢ Execution times within typical ranges, in order for any performance changes to actually be measurable and avoiding iterations running for days

Create a subset

- ➢ Requires process to analyze contents of sample
- ➢ Review coverage and adjust parameters to generate representative tests
- ➢ Select subset that maintains coverage of report types and conditional paths
- ➢ Popularity of individual report types is not a factor at this time



Benchmark Process

A. Perform initial iterations of individual test cases and capture total time for each
B. Evaluate the confidence interval of the average performance for each test case
C. Continue iterations until you achieve the desired confidence interval

Perform initial iterations

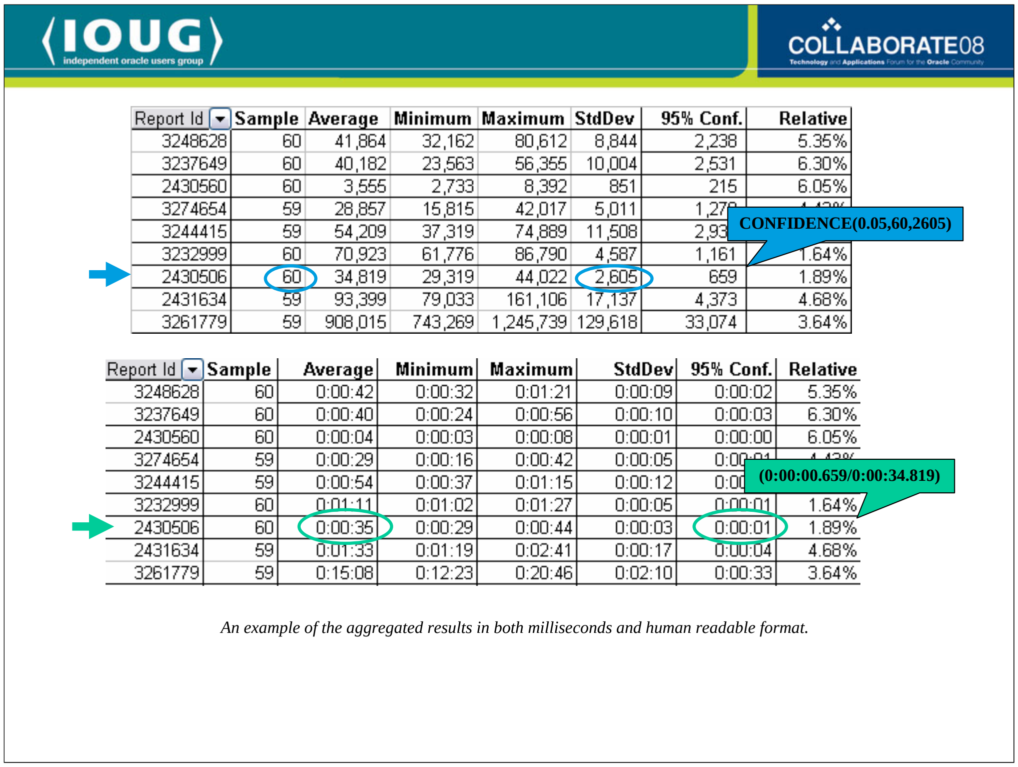
- Run a small number of initial iterations to generate basic data and get a feel for both the average execution time for an iteration and the amount of variance within the individual tests

Evaluate confidence intervals

- Calculate 95% Confidence Interval using the number of executions for each test along with the standard deviation
- Divide this confidence interval by average execution time for the test to provide a relative measure of the size of the confidence interval
- Both values required to determine reliability of average performance estimate, variations of several minutes may be acceptable on large reports running more than an hour while variations of one or two minutes are totally unacceptable on small reports running under two minutes

Continue iterations

- Identify tests without reliable estimate of average performance, in our case where the confidence interval was greater than 30 seconds as long as it was more than 10% of average execution time
- Initially one third of tests were outliers, continue iterations until number of outliers stabilizes the confidence interval on remaining outliers may not converge towards statistically reliable average



Report Id	Sample	Average	Minimum	Maximum	StdDev	95% Conf.	Relative
3248628	60	41,864	32,162	80,612	8,844	2,238	5.35%
3237649	60	40,182	23,563	56,355	10,004	2,531	6.30%
2430560	60	3,555	2,733	8,392	851	215	6.05%
3274654	59	28,857	15,815	42,017	5,011	[illegible]	4.43%
3244415	59	54,209	37,319	74,889	11,508	[illegible]	[illegible]
3232999	60	70,923	61,776	86,790	4,587	1,161	1.64%
2430506	60	34,819	29,319	44,022	2,605	659	1.89%
2431634	59	93,399	79,033	161,106	17,137	4,373	4.68%
3261779	59	908,015	743,269	1,245,739	129,618	33,074	3.64%

CONFIDENCE(0.05,60,2605)

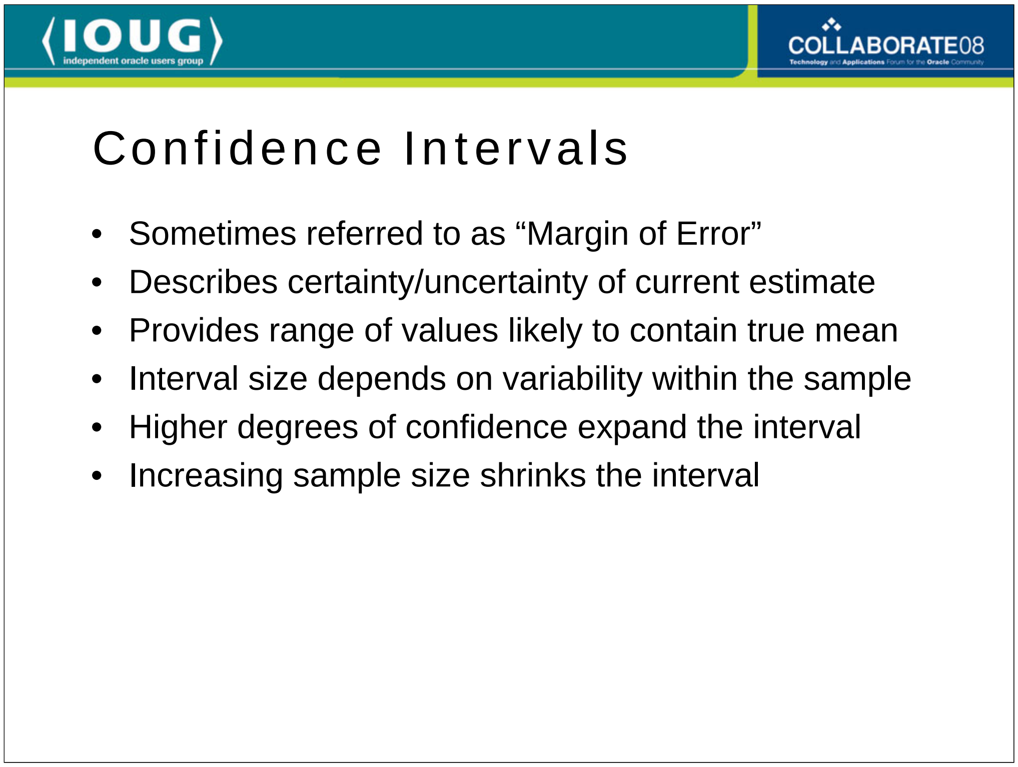
Report Id	Sample	Average	Minimum	Maximum	StdDev	95% Conf.	Relative
3248628	60	0:00:42	0:00:32	0:01:21	0:00:09	0:00:02	5.35%
3237649	60	0:00:40	0:00:24	0:00:56	0:00:10	0:00:03	6.30%
2430560	60	0:00:04	0:00:03	0:00:08	0:00:01	0:00:00	6.05%
3274654	59	0:00:29	0:00:16	0:00:42	0:00:05	0:00:01	4.43%
3244415	59	0:00:54	0:00:37	0:01:15	0:00:12	[illegible]	[illegible]
3232999	60	0:01:11	0:01:02	0:01:27	0:00:05	0:00:01	1.64%
2430506	60	0:00:35	0:00:29	0:00:44	0:00:03	0:00:01	1.89%
2431634	59	0:01:33	0:01:19	0:02:41	0:00:17	0:00:04	4.68%
3261779	59	0:15:08	0:12:23	0:20:46	0:02:10	0:00:33	3.64%

(0:00:00.659/0:00:34.819)

An example of the aggregated results in both milliseconds and human readable format.

Estimating the Average Performance

- The benchmark process produces a log file containing: the timestamp at the start of execution, the elapsed execution time in milliseconds, unique identifying information for the test along with the result code and any error messages
- These results are aggregated for each test across the available iterations calculating: the total number of executions, the standard deviation for the execution time in milliseconds along with the average plus the minimum/maximum values which are used for diagnostic purposes only
- The confidence interval for report [2430506] is .659 seconds, this is determined using the CONFIDENCE() function which accepts the alpha (in our case .05 to produce a 95% confidence interval), along with the sample size (60) and standard deviation (2.605 seconds)
- This means that we are 95% certain that the true mean performance for this report falls between 34.16 and 35.478 seconds, which is the current average plus or minus the confidence interval.
- We then divide this confidence interval (.659 seconds) by current average execution time for the test (34.819 seconds) to provide a relative measure of the size of the confidence interval (in this case the remaining variation is only 1.89% of the currently computed average execution time)


IOUG
independent oracle users group
COLLABORATE08
Technology and Applications Forum for the Oracle Community

Confidence Intervals

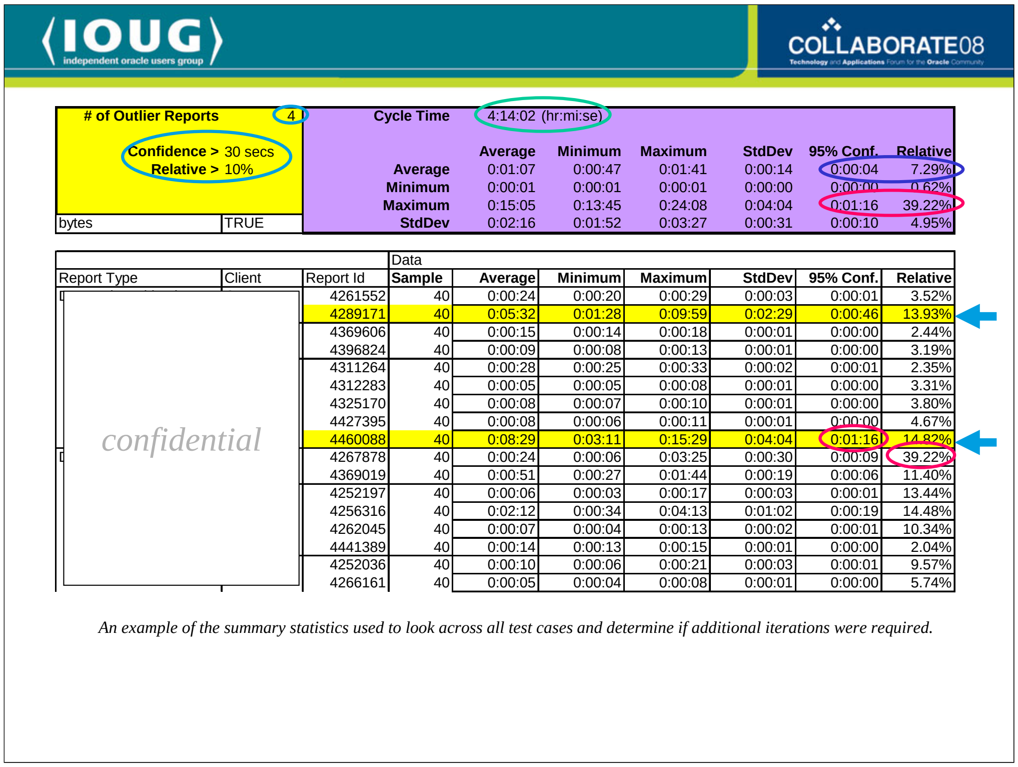
- Sometimes referred to as “Margin of Error”
- Describes certainty/uncertainty of current estimate
- Provides range of values likely to contain true mean
- Interval size depends on variability within the sample
- Higher degrees of confidence expand the interval
- Increasing sample size shrinks the interval

About Confidence Intervals

- Political polls typically report the percentage of likely voters in favor of a particular candidate along with a margin of error for the results
- The statistical method of using random samples to make determinations and inferences about the total population, introduces a level of uncertainty in to the results.
- The goal is for the average from the sample to closely approximate the actual average for the total population from which the sample was obtained (the true mean).
- The confidence interval is the range needed to specify with varying degrees of probability (or confidence) that the current estimate closely approximates the true mean.
- The amount of variability within sample is determined by how much the individual data points differ from each other within the whole population.
- Increased variability means that a larger interval is required, reduced variability means that the interval can be smaller.
- The level of confidence required also dictates the size of the interval, being 70% certain of something requires a much smaller interval than being 90% certain.
- Typically intervals are taking using a 95% level of certainty, meaning that it’s okay to be wrong 5% of the time.
- As the size of the samples get larger, the amount of variability goes down and the interval shrinks while still maintaining the same level of certainty.

Stating your Results

- The mean execution time for this test was 34.819 seconds and we'd expect future samples to fall between 34.16 seconds and 35.478 seconds 95% of the time.
- The mean execution time for this test was 34.819 seconds +/- .659 seconds (with 95% Confidence).
- The mean execution time for this test was 34.819 seconds with a 95% confidence interval of 34.16 seconds to 35.478 seconds



⟨IOUG⟩ independent oracle users group

COLLABORATE08 Technology and Applications Forum for the Oracle Community

# of Outlier Reports 4

Confidence > 30 secs

Relative > 10%

bytes	TRUE

Cycle Time 4:14:02 (hr:mi:se)

	Average	**Minimum**	**Maximum**	**StdDev**	**95% Conf.**	**Relative**
Average	0:01:07	0:00:47	0:01:41	0:00:14	0:00:04	7.29%
Minimum	0:00:01	0:00:01	0:00:01	0:00:00	0:00:00	0.62%
Maximum	0:15:05	0:13:45	0:24:08	0:04:04	0:01:16	39.22%
StdDev	0:02:16	0:01:52	0:03:27	0:00:31	0:00:10	4.95%

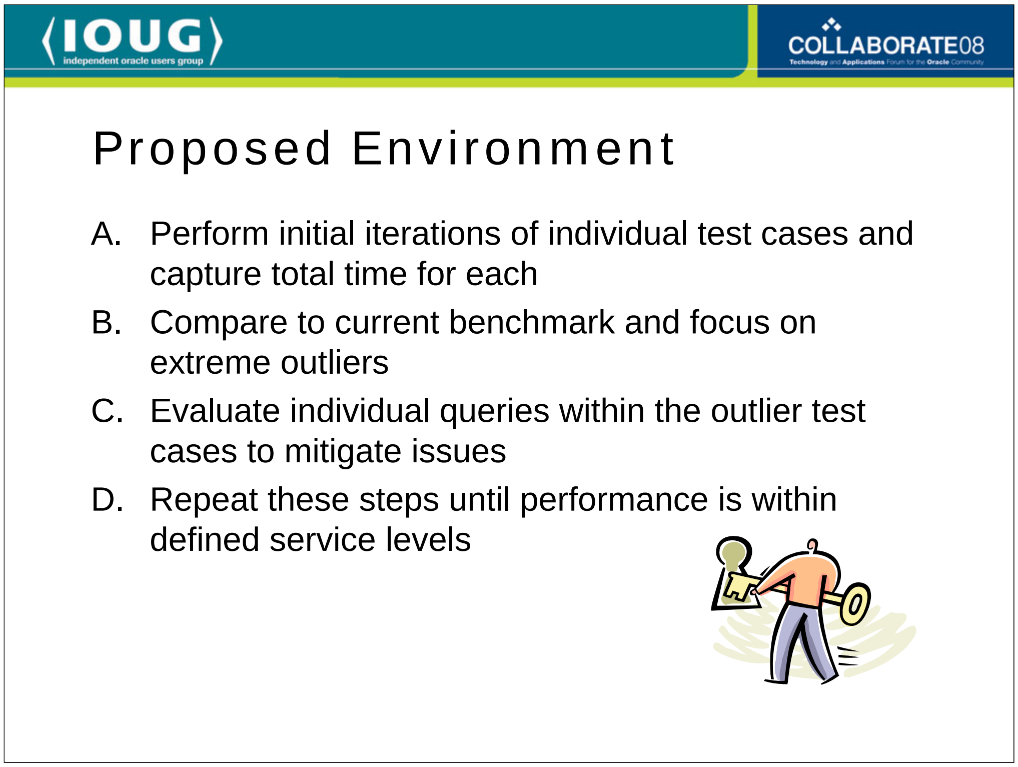
confidential

			Data						
Report Type	Client	Report Id	**Sample**	**Average**	**Minimum**	**Maximum**	**StdDev**	**95% Conf.**	**Relative**
[redacted]	[redacted]	4261552	40	0:00:24	0:00:20	0:00:29	0:00:03	0:00:01	3.52%
[redacted]	[redacted]	4289171	40	0:05:32	0:01:28	0:09:59	0:02:29	0:00:46	13.93%
[redacted]	[redacted]	4369606	40	0:00:15	0:00:14	0:00:18	0:00:01	0:00:00	2.44%
[redacted]	[redacted]	4396824	40	0:00:09	0:00:08	0:00:13	0:00:01	0:00:00	3.19%
[redacted]	[redacted]	4311264	40	0:00:28	0:00:25	0:00:33	0:00:02	0:00:01	2.35%
[redacted]	[redacted]	4312283	40	0:00:05	0:00:05	0:00:08	0:00:01	0:00:00	3.31%
[redacted]	[redacted]	4325170	40	0:00:08	0:00:07	0:00:10	0:00:01	0:00:00	3.80%
[redacted]	[redacted]	4427395	40	0:00:08	0:00:06	0:00:11	0:00:01	0:00:00	4.67%
[redacted]	[redacted]	4460088	40	0:08:29	0:03:11	0:15:29	0:04:04	0:01:16	14.82%
[redacted]	[redacted]	4267878	40	0:00:24	0:00:06	0:03:25	0:00:30	0:00:09	39.22%
[redacted]	[redacted]	4369019	40	0:00:51	0:00:27	0:01:44	0:00:19	0:00:06	11.40%
[redacted]	[redacted]	4252197	40	0:00:06	0:00:03	0:00:17	0:00:03	0:00:01	13.44%
[redacted]	[redacted]	4256316	40	0:02:12	0:00:34	0:04:13	0:01:02	0:00:19	14.48%
[redacted]	[redacted]	4262045	40	0:00:07	0:00:04	0:00:13	0:00:02	0:00:01	10.34%
[redacted]	[redacted]	4441389	40	0:00:14	0:00:13	0:00:15	0:00:01	0:00:00	2.04%
[redacted]	[redacted]	4252036	40	0:00:10	0:00:06	0:00:21	0:00:03	0:00:01	9.57%
[redacted]	[redacted]	4266161	40	0:00:05	0:00:04	0:00:08	0:00:01	0:00:00	5.74%

An example of the summary statistics used to look across all test cases and determine if additional iterations were required.

Evaluating the Benchmark

- To determine the health of a benchmark, you need to identify the number of test cases which are outside of your acceptable range of confidence intervals.
- The total of the average execution time for all reports is the expected cycle time for each iteration
- The average and maximum values for the Confidence Interval and Relative change can be used as a diagnostic on the size of the intervals are on the remaining outliers.
- The maximum value may not actually represent an issue based on the combination of the two thresholds.
- Eventually the number of remaining outliers will either work themselves out with increased iterations or stabilize at a small number which require more iterations than you are willing to wait for.



Proposed Environment

A. Perform initial iterations of individual test cases and capture total time for each
B. Compare to current benchmark and focus on extreme outliers
C. Evaluate individual queries within the outlier test cases to mitigate issues
D. Repeat these steps until performance is within defined service levels

Perform initial iterations

- Perform as few iterations as possible
- Shake out worst performance issues
- Aggregates calculated for each test are identical to benchmark set

Compare to current benchmark

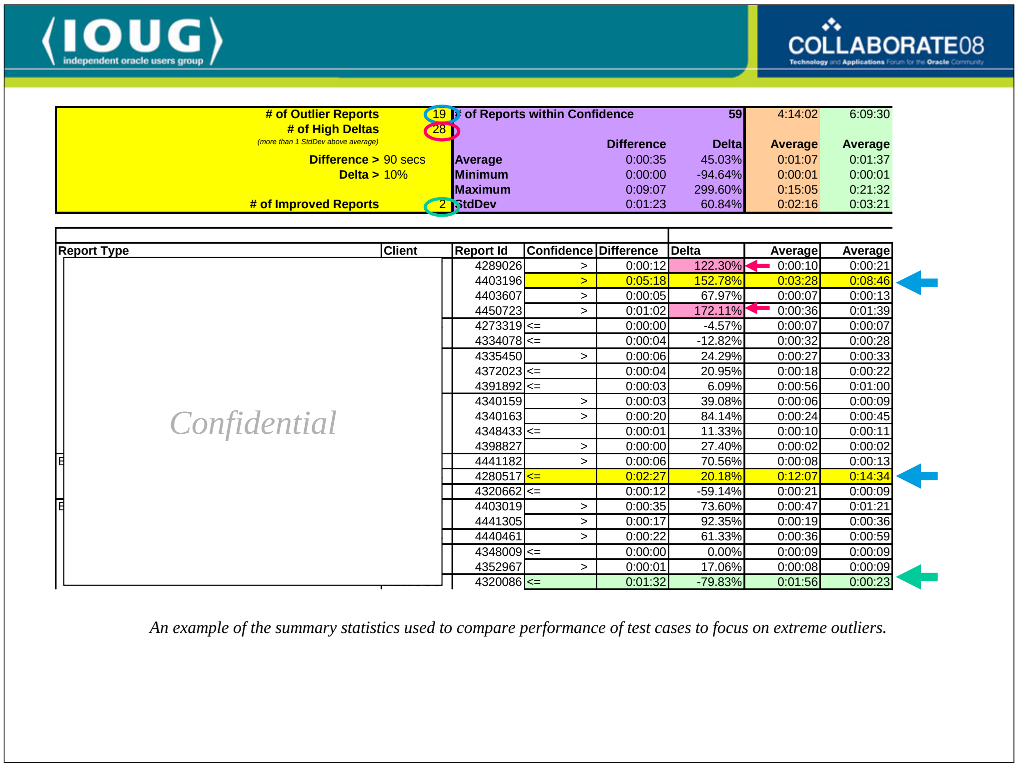
- Compute actual difference in average execution times along with delta
- Focus on most extreme differences in performance
- Continue lowering the threshold to highlight the dozen tests with worst performance

Evaluate individual queries

- Execute the test case again in both architectures and generate detailed log showing execution time for each SQL statement
- Review logs side by side, calculating difference in time for each statement
- Analyze execution plans and actual performance for long running statements
- Identify optimizer hints or other changes required to achieve comparable performance

Repeat these steps

- Keep looping until performance is equivalent across both architectures
- Avoid temptation to focus remaining iterations on specific tests
- Measure impact of tuning across application, since fixes for one test could correct issues within other tests, or it could cause other issues
- Frequent executions with updated code are essential



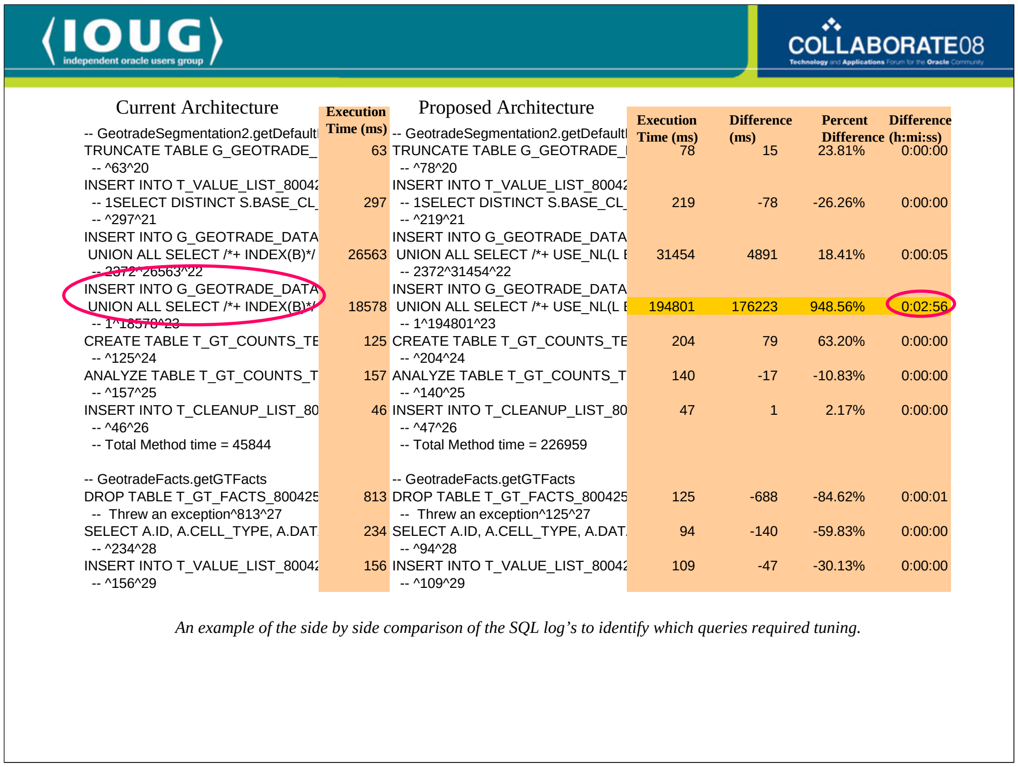
# of Outlier Reports	19	**# of Reports within Confidence**		**59**	4:14:02	6:09:30
# of High Deltas *(more than 1 StdDev above average)*	28					
			Difference	**Delta**	**Average**	**Average**
Difference > 90 secs		**Average**	0:00:35	45.03%	0:01:07	0:01:37
Delta > 10%		**Minimum**	0:00:00	-94.64%	0:00:01	0:00:01
		Maximum	0:09:07	299.60%	0:15:05	0:21:32
# of Improved Reports	2	**StdDev**	0:01:23	60.84%	0:02:16	0:03:21

Report Type	Client	Report Id	Confidence	Difference	Delta	Average	Average
Confidential		4289026	>	0:00:12	122.30%	0:00:10	0:00:21
		4403196	>	0:05:18	152.78%	0:03:28	0:08:46
		4403607	>	0:00:05	67.97%	0:00:07	0:00:13
		4450723	>	0:01:02	172.11%	0:00:36	0:01:39
		4273319	<=	0:00:00	-4.57%	0:00:07	0:00:07
		4334078	<=	0:00:04	-12.82%	0:00:32	0:00:28
		4335450	>	0:00:06	24.29%	0:00:27	0:00:33
		4372023	<=	0:00:04	20.95%	0:00:18	0:00:22
		4391892	<=	0:00:03	6.09%	0:00:56	0:01:00
		4340159	>	0:00:03	39.08%	0:00:06	0:00:09
		4340163	>	0:00:20	84.14%	0:00:24	0:00:45
		4348433	<=	0:00:01	11.33%	0:00:10	0:00:11
		4398827	>	0:00:00	27.40%	0:00:02	0:00:02
		4441182	>	0:00:06	70.56%	0:00:08	0:00:13
		4280517	<=	0:02:27	20.18%	0:12:07	0:14:34
		4320662	<=	0:00:12	-59.14%	0:00:21	0:00:09
		4403019	>	0:00:35	73.60%	0:00:47	0:01:21
		4441305	>	0:00:17	92.35%	0:00:19	0:00:36
		4440461	>	0:00:22	61.33%	0:00:36	0:00:59
		4348009	<=	0:00:00	0.00%	0:00:09	0:00:09
		4352967	>	0:00:01	17.06%	0:00:08	0:00:09
		4320086	<=	0:01:32	-79.83%	0:01:56	0:00:23

An example of the summary statistics used to compare performance of test cases to focus on extreme outliers.

Evaluating the Benchmark

- To evaluate the architecture change, you need to review the differences in average performance across both benchmarks.
- Set thresholds in order to focus on most extreme differences in performance
- Like the confidence intervals, you need to compare both the actual difference in average execution times along with delta or relative change.
- As you tune the problems out of the system, continue lowering the threshold to highlight the dozen tests with worst performance
- Near the end of the process, review tests where delta is more than one standard deviation above average delta to find extreme differences in performance.
- Compare both confidence intervals to determine if interval from proposed environment is less than or equal to interval from current environment
- Eventually you will run into performance differences that you cannot tune out of the new environment.

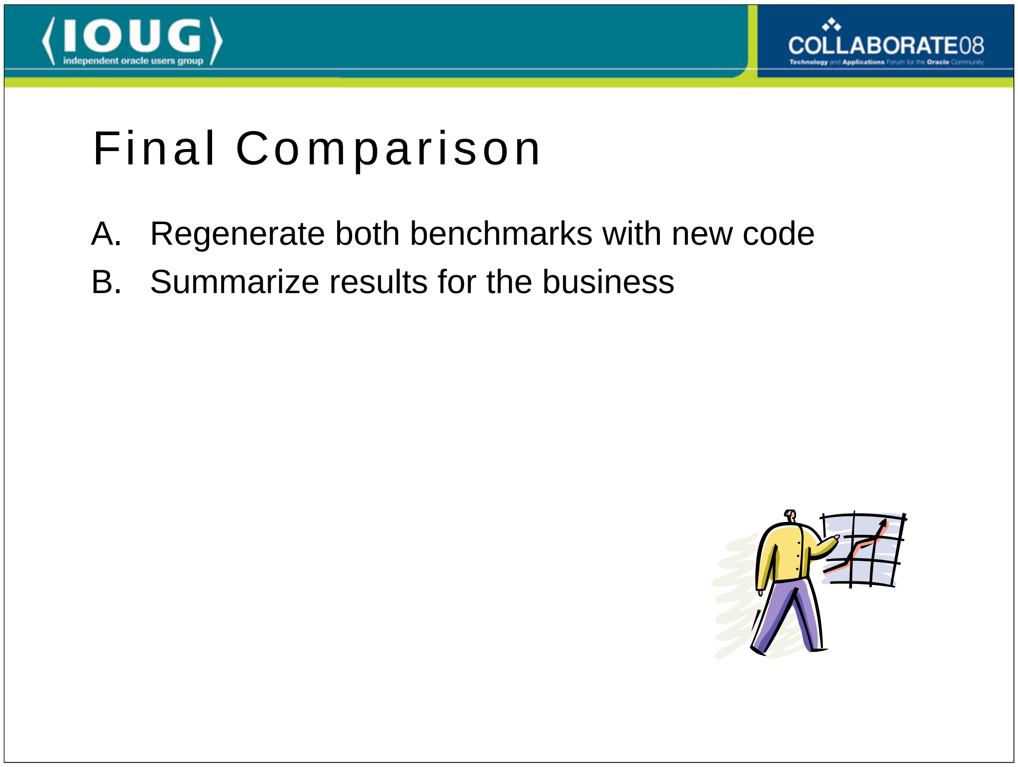

IOUG
independent oracle users group
COLLABORATE08
Technology and Applications Forum for the Oracle Community

Current Architecture	Execution Time (ms)	Proposed Architecture	Execution Time (ms)	Difference (ms)	Percent Difference	Difference (h:mi:ss)
-- GeotradeSegmentation2.getDefault		-- GeotradeSegmentation2.getDefault				
TRUNCATE TABLE G_GEOTRADE_ -- ^63^20	63	TRUNCATE TABLE G_GEOTRADE_ -- ^78^20	78	15	23.81%	0:00:00
INSERT INTO T_VALUE_LIST_80042 -- 1SELECT DISTINCT S.BASE_CL_ -- ^297^21	297	INSERT INTO T_VALUE_LIST_80042 -- 1SELECT DISTINCT S.BASE_CL_ -- ^219^21	219	-78	-26.26%	0:00:00
INSERT INTO G_GEOTRADE_DATA UNION ALL SELECT /*+ INDEX(B)*/ -- 2372^26563^22	26563	INSERT INTO G_GEOTRADE_DATA UNION ALL SELECT /*+ USE_NL(L -- 2372^31454^22	31454	4891	18.41%	0:00:05
INSERT INTO G_GEOTRADE_DATA UNION ALL SELECT /*+ INDEX(B)*/ -- 1^18578^23	18578	INSERT INTO G_GEOTRADE_DATA UNION ALL SELECT /*+ USE_NL(L -- 1^194801^23	194801	176223	948.56%	0:02:56
CREATE TABLE T_GT_COUNTS_TE -- ^125^24	125	CREATE TABLE T_GT_COUNTS_TE -- ^204^24	204	79	63.20%	0:00:00
ANALYZE TABLE T_GT_COUNTS_T -- ^157^25	157	ANALYZE TABLE T_GT_COUNTS_T -- ^140^25	140	-17	-10.83%	0:00:00
INSERT INTO T_CLEANUP_LIST_80 -- ^46^26 -- Total Method time = 45844	46	INSERT INTO T_CLEANUP_LIST_80 -- ^47^26 -- Total Method time = 226959	47	1	2.17%	0:00:00
-- GeotradeFacts.getGTFacts		-- GeotradeFacts.getGTFacts				
DROP TABLE T_GT_FACTS_800425 -- Threw an exception^813^27	813	DROP TABLE T_GT_FACTS_800425 -- Threw an exception^125^27	125	-688	-84.62%	0:00:01
SELECT A.ID, A.CELL_TYPE, A.DAT -- ^234^28	234	SELECT A.ID, A.CELL_TYPE, A.DAT -- ^94^28	94	-140	-59.83%	0:00:00
INSERT INTO T_VALUE_LIST_80042 -- ^156^29	156	INSERT INTO T_VALUE_LIST_80042 -- ^109^29	109	-47	-30.13%	0:00:00

An example of the side by side comparison of the SQL log's to identify which queries required tuning.

Working with Individual Test Cases

- After pasting the logs in from both environments, there were formulas in the tan columns which extracted the execution time for each step and given times in both environments calculated the difference in performance
- Look for the most extreme differences within the test case and focus on running that SQL again in both environments
- Sometimes you will need to grab some of the previous statements to set up the environment for your test
- Compare the execution plans in both environments and add hints as necessary to bring the plans in line, which hopefully also mitigates the performance issue.



Final Comparison

A. Regenerate both benchmarks with new code
B. Summarize results for the business

Regenerate both benchmarks

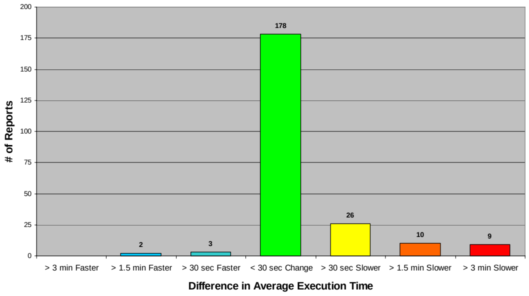
- Best to repeat your initial benchmark with latest code
- Strengthen benchmark in proposed environment
- Reliable estimates of average performance are essential for final comparisons
- Proceed once tests in both environments fall within desired confidence interval

Summarize results

- Most difficult part of entire process
- Need to summarize performance differences for review and approval by stakeholders
- Exposing risks associated with architecture change
- Create a normal distribution by collapsing tests into ranges by both actual difference and percentage change
- Group the individual tests by report type based on actual usage
- Show that worst performing tests are not indicative of a general problem within a specific class of test cases.

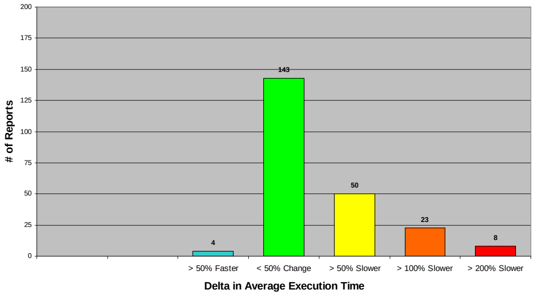
Final Results: Actual Difference in Average Execution Time

- Over 92% of the benchmark reports run 1½ min slower or less on average, with 80% of the reports showing no detectable difference (or an improvement) in performance.
- The execution plan hints to mitigate the remaining performance differences resulted in much worse performance under the other prompt selections and were rolled back.



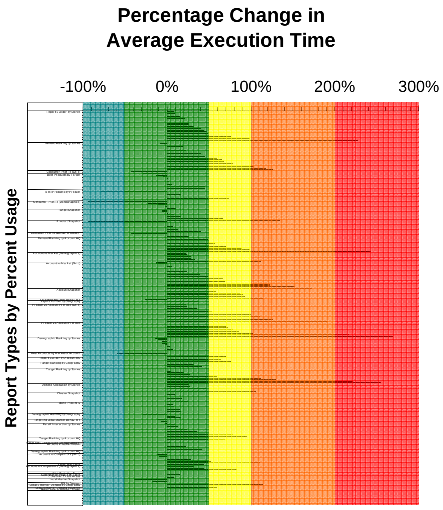
Final Results: Percentage Change in Average Execution Time

- Over 86% of the benchmark reports were 100% slower or less on average, with 64% of the reports showing no detectable difference (or an improvement) in performance.
- The average difference in execution time is only 48 seconds for the reports with a 200% increase or more and under 2 minutes for the reports with 100% to 200% increase, in comparison, the average improvement is just over 1 minute.



Performance across Report Types

- Looking at all 228 individual prompt sets across the 43 report types tested; 64% are contained within the green or blue zones.
- The average difference in execution time is only 48 seconds for the 8 entries stretching into the red zone.
- The average difference in execution time is under 2 minutes for the 23 entries within the orange zone.
- The average improvement in execution time just over 1 minute for the 4 entries in the blue zone.
- The 9 remaining outliers running 3 minutes longer or more under the DB links are split evenly within the yellow and orange zones with an average percentage change of just over 100%.



Items Learned in this Session

- Insight into the issues with testing application performance on a complex decision support system.
- A method for capturing real-world test cases to insure the results are relevant to the business.
- Examples of a proven statistics based approach that compares performance before/after a change.

Questions?

David A Haas
Senior Director, Database Architecture
Consumer Panel Services
The Nielsen Company
39 East Chestnut Street
Lancaster, PA 17602-2701
717-397-1500
dave.haas@nielsen.com

Thank You

- Please complete your evaluation form
  - ARE WE DONE YET?
  - CERTIFYING APPLICATION PERFORMANCE UNDER NEW DATABASE VERSIONS OR ARCHITECTURES
  - David A Haas
  - Session #421

nielsen