SARD: A Statistical Approach for Ranking Database Tuning Parameters

Biplob K. Debnath [#1], David J. Lilja [#2], Mohamed F. Mokbel [*3]

[#]*Department of Electrical and Computer Engineering*
University of Minnesota, Twin Cities, USA.
[1]debna004@umn.edu
[2]lilja@ece.umn.edu

[*]*Department of Computer Science*
University of Minnesota, Twin Cities, USA.
[3]mokbel@cs.umn.edu

Abstract

Traditionally, DBMSs are shipped with hundreds of configuration parameters. Since the database performance highly depends on the appropriate settings of the configuration parameters, DBAs spend a lot of their time and effort to find the best parameter values for tuning the performance of the application of interest. In many cases, they rely on their experience and some rules of thumbs. However, time and effort may be wasted by tuning those parameters which may have no or marginal effects. Moreover, tuning effects also vary depending on the expertise of the DBAs, but skilled DBAs are increasingly becoming rare and expensive to employ. To address these problems, we present a Statistical Approach for Ranking Database parameters (SARD), which is based on the Plackett & Burman statistical design methodology. SARD takes the query workload and the number of configuration parameters as inputs, and using only a linear number of experiments, generates a ranking of database parameters based on their relative impacts on the DBMS performance. Preliminary experimental results using TPC-H and PostgreSQL show that SARD generated ranking can correctly identify critical configuration parameters.

I. Introduction

Businesses are increasingly building larger databases to cope with the rapid current growth of data. Consistent performance of the underlying database system is a key to success of a business. A typical database management system (DBMS) has hundreds of configuration parameters and the appropriate setting of these parameters play a critical role in performance. DBMSs are shipped with default values for all configuration parameters targeting a wide range of applications. Although some utilities exist (for example, DB2's `autoconfigure`) to recommend values of the some configuration parameters, but the suggested values are not always accurate as the recommendation is based on query workload's generic characteristics. Database administrators (DBAs) are expected to tune the performance of the shipped DBMSs to the application of their interest. The success of tuning depends on many factors including the query workload, relational schemas, as well as the expertise of the DBAs [1]. However, skilled DBAs are becoming increasingly rare and expensive [2]. A recent study on information technology versus DBA costs showed that personnel cost is the largest category of the total cost, estimated at 47% of the total cost of ownership [3]. Many DBAs spend nearly a quarter of their time on tuning activities [1]. To reduce the total cost of ownership, it is of essence that DBAs focus only on tuning those configuration parameters which have the most impact on system performance.

A sound statistical methodology for quantifying the impact of each configuration parameter and the interactions among those parameters on a DBMS performance is to perform a *full factorial design*, where every combination of input values of the configuration parameters are considered. However, the major problem in applying a *full factorial design* in a DBMS is the large number of configuration parameters. For example, PostgreSQL [4] has approximately 100 configuration parameters and all

parameters have multiple possible values. Given a query workload, even if each configuration parameter assumes only two values, we have to perform 2^{100} experiments to apply a *full factorial design* for each query of the workload, which is not feasible in terms of time and effort. To avoid this problem, in many cases DBAs rely on their experience and rules of thumb to select the appropriate configuration parameters for tuning. Nonetheless, as heuristics out of experiences and intuition are often used, time and effort may be wasted to enhance the performance by tuning those parameters which may have no or marginal effects on overall performance. Misdirected tuning efforts increase the total cost of ownership [5]–[8]. A ranking of the parameters based on their impact on system performance will greatly help DBAs to prioritize their tuning tasks. To the best of our knowledge, there is no study which statistically provides a ranking of the configuration parameters based on their impact on DBMS performance.

In this paper, using a *design of experiments*-based PLACKETT & BURMAN (P&B) methodology [9], we present a Statistical Approach for Ranking Database configuration parameters (SARD) based on their impact on the DBMS performance. In particular, SARD addresses the following problem: *Given a DBMS, a set of configuration parameters, a range of values for all parameters, and a query workload; find a relative ranking of the parameters based on their impact on performance.* A workload can be a set of benchmark queries, for example, TPC queries, or can be a set of data manipulation language (DML) statements collected over a fixed amount of time using profiling tools. SARD can be used to discard unimportant tuning parameters which have marginal or no impact on DBMS performance. Our objective is to find a solution which is feasible in terms of time and effort.

The main idea of SARD is to conduct a set of experiments that provide an approximate sampling of the entire search space. In each experiment, parameter values are varied systematically over a specified range of acceptable values. Each experiment indicates what will be the response time a query, if a specific combination of configuration parameter values are used to set up a database server. Subsequent analysis of the collected experimental data are used to estimate the effects of configuration parameters on system performance. To reduce the exponential number of experiments required for applying a *full factorial design*, SARD uses the P&B *two-level factorial design* methodology based on the following assumptions: 1) stimulating the system with monotonically non-decreasing parameters at their extreme values will provoke the greatest response for each parameter; and 2) only single and two-factor parameters interactions need to be considered. Adopting P&B design method helps us to reduce the required number of experiments from exponential to linear.

Our contribution in this paper is mainly *a methodology for ranking database configuration parameters* based on their impact on DBMS performance for a given query workload. Preliminary experimental evaluation shows that SARD is able to find performance bottleneck parameters using the PostgreSQL [4] and the TPC-H benchmark [10].

The remainder of the paper is organized as follows: Section II describes our design of experiments-based methodology and a brief overview of the statistical P&B design. Section III describes the phases of SARD in detail. Section IV describes the experimental setup. Section V explains our results. Section VI describes the related work on DBMS performance tuning. Finally, Section VII concludes the discussion and lists some future extensions.

II. Design of Experiments Based Methodology

SARD is a *design of experiments*-based approach. The major goal of the *design of experiments* is to gather the maximum information about a system with minimum effort [11]. Experiments are conducted based on a given specification to collect information about system performance. The subsequent analysis of resulting experimental data is used to identify the important factors (i.e, parameters), and the presence of interactions between the factors. The simplest design strategy to quantify the impact of all factors and interactions is to apply a *full factorial design*, for example ANOVA, in which system response is measured for all possible input combinations [11]. However, it requires an exponential number of experiments in the number of parameters.

To reduce the number of experiments, SARD makes a few assumptions. First, for each parameter SARD considers only two values: minimum and maximum. The intuition behind this is that stimulating the system with inputs at their extreme values will provoke the maximum range of output responses for each input. A second related assumption is that the provoked response, such as the total execution time, is a monotonic function of the input parameter values. The third assumption is based on *sparsity of effects principle*: system response is largely dominated by a few main factors and low-order interactions; the effect of higher order interactions on response is not statistically significant. As a consequence, we can safely ignore the effects of higher order interactions.

Based on these assumptions, SARD uses a *two-level factorial* design named Plackett & Burman (P&B) de-

sign [9], which requires only linear number of experiments compared to the exponential number of experiments required by the *full-factorial design.* For each experiment of the P&B design, the value of each parameter is given by prescribed *design matrix*. Table I gives an example of the *design matrix*, depicted by the columns 2-8. Each row of the matrix corresponds to one experiment. Each cell in the matrix indicates the value to be used for a parameter in the corresponding experiment. Entry in the matrix is either "+1" or "-1". "+1" corresponds to a value slightly higher than the normal range of values for that parameter and "-1" corresponds to a value slightly lower than the normal range of that parameter. The "+1" and "-1" values are not restricted to only numeric values. For example, for the buffer page replacement algorithm, the "-1" value can be "RANDOM" and "+1" value can be "CLOCK". Experiments are conducted by setting up the values according to the design matrix, and query execution times are recorded, as in the last column of Table I. The net effect of each parameter is estimated by multiplying the response value with the corresponding "+1" or "-1" for each row and summing the values across all rows. The absolute value of net effect is used to determine the relative importance of that parameter.

	A	B	C	D	E	F	G	Execution Time		
								Q1	Q2	Q3
R1	+1	+1	+1	-1	+1	-1	-1	34	110	10.2
R2	-1	+1	+1	+1	-1	+1	-1	19	72	10.1
R3	-1	-1	+1	+1	+1	-1	+1	111	89	10.3
R4	+1	-1	-1	+1	+1	+1	-1	37	41	10.3
R5	-1	+1	-1	-1	+1	+1	+1	61	96	10.2
R6	+1	-1	+1	-1	-1	+1	+1	29	57	10.2
R7	+1	+1	-1	+1	-1	-1	+1	79	131	10.3
R8	-1	-1	-1	-1	-1	-1	-1	19	47	10.1
R9	-1	-1	-1	+1	-1	+1	+1	135	107	10.3
R10	+1	-1	-1	-1	+1	-1	+1	56	74	10.3
R11	+1	+1	-1	-1	-1	+1	-1	112	48	10.1
R12	-1	+1	+1	-1	-1	-1	+1	74	91	10.1
R13	+1	-1	+1	+1	-1	-1	-1	55	99	10.3
R14	-1	+1	-1	+1	+1	-1	-1	117	123	10.1
R15	-1	-1	+1	-1	+1	+1	-1	51	77	10.3
R16	+1	+1	+1	+1	+1	+1	+1	76	81	10.2

TABLE I. Columns 2-8 and rows R1-R16 form the P&B design matrix for the seven parameters A, B, C, D, E, F, and G. Last three columns contain the execution time of the queries Q1, Q2, and Q3. Rows R1-R8 contain the base P&B design matrix, and rows R9-R16 are needed if *foldover* is used.

III. SARD: a P&B Design for DBMS

SARD uses P&B design methodology to estimate the effects of configuration parameters on DBMS performance. SARD has three major phases. In the first phase, SARD estimates the P&B effect of each configuration parameter on the DBMS performance for each query of the workload. In the second phase, for each query, the configuration parameters are ranked based on the relative magnitude of the P&B effect. In the third phase, the rankings of the configuration parameters for all individual queries are combined to determine the final ranking of the parameters for the entire query workload. For illustration, we will use Table I as a running example, where seven configuration parameters A, B, C, D, E, F, and G need to be ranked. The workload consists of three queries: Q1, Q2, and Q3. The last three columns correspond to the execution times of the query Q1, Q2, and Q3. Now, we will discuss the three phases of SARD in detail in the following subsections.

A. Phase I: Parameter Effect Estimation

At first, a *P&B design matrix* is constructed, which gives the specification of values used for each parameter in each experiment. The dimension of design matrix depends on the number of configuration parameters, N. The base design matrix has X rows and $X-1$ columns, where X is the next multiple of 4 greater than N, i.e., $X = (floor(N/4)+1)*4$. For example, if $N=5$, then $X=8$, while if $N=8$, then $X=12$. The value of X indicates the number of experiments that need to be conducted in SARD to collect data for estimating the P&B effect. In the design of optimal multifactorial experiments work [9], Plackett and Burman recommended the parameter values setting,"+1" or "-1", for $X = 8, 12, 16, \ldots, 96, 100$ experiments. SARD sets the first row of the *P&B design matrix* based on that recommendations according to the value of X. Rest of the $(X-1)$ rows of the *P&B design matrix* are constructed by right cyclic shifting of the immediate preceding row. All entries of the X-th row of the P&B design matrix are set to "-1". The columns 2-8 in the first eight rows (R1-R8) of the Table I gives a base P&B design matrix for $N=7$.

An improvement that increases the accuracy of the base P&B design is the P&B design with *foldover* [12], which requires additional X experiments. These additional rows are constructed by reversing the sign of the top X rows entries. The $(X+i)$-th row is formed by reversing the sign of the i-th row. The columns 2-8 in the last eight rows (R9-R16) of Table I gives additional design matrix entries for the *foldover* for $N=7$. If $N < (X-1)$, means that number of columns in the P&B design matrix

is more than the number of configuration parameters. In this case, the additional $(X - N - 1)$ last columns of the P&B matrix are considered as dummies, are simply ignored. For each query of the workload, the i-th experiment is conducted by setting the parameters values of the configuration parameters according to the i-th row of the P&B design matrix. The effect of the each configuration parameter is calculated by multiplying the corresponding "+1" or "-1" of that parameter in the i-th row of the P&B matrix with the query execution time, and summing up the products across all rows of the design matrix.

For the illustrative example, in Table I, N is seven and the value of X is eight. For the base case, we need to conduct eight experiments. If *foldover* is used, we need to conduct 16 experiments. In this paper, we assume that *foldover* is used. The specification of the parameter values need to be used in all 16 experiments are given in the columns 2-8 and rows R1-R16 of Table I. The first row (R1) of the *P&B design matrix* in Table I, is copied from [9] according to the value of $X = 8$. For query Q1, 16 experiments are conducted and execution time is recorded, as shown in the ninth column of Table I. Similarly, the execution time of the queries Q2 and Q3 are recorded in the 10-th and 11-th column. Now, the net effect of the first parameter A for the query Q1 is calculated by multiplying the entries in the second column with entries in the ninth column and summing up across all 16 rows (R1-R16). For query Q1, the net effect of the parameter A is estimated as:

$Effect_A = abs((+1 * 34) + (-1 * 19) + \ldots + (-1 * 51) + (+1 * 76)) = abs(-109) = 109.$

Similarly, the net effect of the second parameter B for query Q1 is calculated by multiplying the entries in the third column with the entries in the ninth column and summing across all 16 rows (R1-R16). The net effect of A for query Q2 is calculated by multiplying the entries in the second column with entries in the tenth column and summing across all 16 rows (R1-R16), and so on. The net P&B effects of all seven parameters for the queries Q1, Q2, and Q3 are shown in Table II.

SARD can also determine the sensitivity of a query to parameter tuning. For each query, SARD calculates the *standard deviation* of the net effects for the all configuration parameters. No matter how large the P&B effects are, if the standard deviation of the effects is very low, this means that all effects are virtually similar, the query performance will not be affected by the change in configuration parameter settings. In this case, SARD ignores the ranking. The standard deviation of net P&B effects of the parameters for queries Q1, Q2, and Q3 are 136.4, 123.3, and 0.44, respectively as listed in the last column of Table II. However, as the standard deviation of the P&B effects of the configuration parameters for query Q3 is very small, SARD ignores the rankings of the parameters for query Q3.

	A	B	C	D	E	F	G	stdev
Q1	109	79	167	193	21	25	177	136.4
Q2	61	161	9	143	39	185	109	123.3
Q3	0.40	0.80	0.00	0.40	0.40	0.00	0.40	0.44

TABLE II. The P&B effects for the queries Q1, Q2, and Q3.

	A	B	C	D	E	F	G
Q1	0.6	0.4	0.9	1	0.1	0.1	0.9
Q2	0.3	0.9	0.0	0.8	0.2	1.0	0.6

TABLE III. The P&B normalized effects with respect to the maximum effect for the queries Q1 and Q2.

1) Some Discussions: SARD assumes that query execution time is a monotonic function of values for a parameter, therefore the high and low values of a parameter must be picked carefully. To find out (a) whether the execution time is a monotonic function of values for a parameter, and (b) the high and low values, such that execution time is a monotonic function of values in that range, either experience or expertise is required, or more experiments must be conducted. Given the large number of parameters, and the large number of possible values for each parameter, this is a non-trivial problem. Therefore like others tool, the SARD system most likely needs to be provided by the DBMS vendor, and cannot be used as a standalone third-party tool that can be easily adapted for different DBMSs. This is one of our future work, to make SARD a standalone tool.

SARD assumes that there is little interaction among the parameters. Right now, SARD can correctly take care of first and second order interactions using P&B design with *foldover*. For a DBMS, where there are a large number of parameters, however, this assumption does not always hold. So still DBAs' expertise are needed to justify the SARD results. Quantifying the impact of this assumption is an item for future work on SARD.

B. Phase II: Parameter Ranking for a Query

Once the effects of all configuration parameters and the sensitivity of a query is determined, the next step is to rank parameters. If it is insensitive to parameter changes, the ranking can safely be ignored; the performance of

Normalized to Max Effect							
	A	**B**	**C**	**D**	**E**	**F**	**G**
Q1	4	5	3	1	7	7	3
Q2	5	2	7	3	6	1	4

TABLE IV. Ranking of the configuration parameters for the queries Q1 and Q2. Query Q3 is not included as it is not sensitive to tuning.

an insensitive query is unaffected by parameter values changes.

To rank the configuration parameters, we can simply order them based on the descending order of magnitude of the P&B effects. However, problems can arise if some effects are very close to each other. Intuitively, this situation means that the corresponding parameters' effects are virtually all the same, but the sorting method assigned them to a different ranking order. For example, there are four parameters P, Q, R, and S and P&B effects are 1500, 50.6, 51.4, and 3000, respectively. The ranking due to sorting method will be 2, 4, 3, and 1. However, the effects of parameters Q and R are very close. Since the effects are similar, they should be assigned to the same rank. To avoid this problem of the simple sorting method, effects are normalized with respect to the maximum effect, rounded to the first decimal point, and sorted in descending order. All the parameters with the same normalized effect are assigned the same rank. For example, in the previous example the normalized effects of parameters P, Q, R, and S are 0.5, 0.0, 0.0, and 1.0, respectively. According to the rounding method the ranks are 2, 4, 4, and 1, respectively. In the continuing example, the normalized P&B effect for the queries Q1 and Q2 is listed in Table III and ranking due to rounding is listed in Table IV.

C. Phase III: Parameter Ranking for the Workload

After determining the ranking of the parameters for each individual query, next step is to estimate the ranking of parameters for the entire workload. The queries which are insensitive to parameter tuning, are not included in workload ranking calculation.

To estimate the rank of the configuration parameters for the the entire workload, ranks are summed across all queries, averaged and sorted in ascending order. The most important parameters will have the lowest cumulative rank. For example, in Table IV, for the entire workload, the average rankings of the parameters A, B, C, D, E, F, and G are 4.5, 3.5, 5.0, 2.0, 6.5, 4.0, and 3.5, respectively. The final ranking for the parameters of the workload in Table I will be 5, 3, 6, 1, 7, 4, and 3, respectively. The ranking indicates that P_4 is the most important parameter, P_2 and P_7 are the second most important parameters, followed by P_6, P_1, P_3, and P_5 in order.

IV. Experimental Setup

In this section, we will give an overview of the DBMS parameters, workload, and the machine setup used for collecting experimental data. We have conducted all experiments in a machine with two Intel XEON 2.0 GHz w/HT CPUs, 2 GB RAM, and 74 GB 10,000 RPM disk.

A. Workload

For demonstration, we have used a workload consisting of five TPC-H queries, {Q1, Q8, Q9, Q13, Q16}. TPC-H database is populated by data generation program ***dbgen*** with *scaling factor* (SF) of 1 (i.e., data size is 1GB). Queries are generated by ***qgen*** program supplied with TPC-H benchmark. For collecting data, queries are modified by adding EXPLAIN ANALYZE. Queries are executed one at a time. More detail experimental results using a different workload and full TPC-H queries can be found in [13].

B. Parameters Considered

For demonstration, we have used the PostgreSql8.2, which has approximately 100 configuration parameters [4]. As our queries are read-only, many of the parameters are not relevant. We have considered only those parameters which seem to be relevant to read only queries. We have also deliberately selected parameters `checkpoint_timeout` and `fsync` which do not have any effect on read only queries. Intuitively the relative impact of this type of parameters should be low or zero for read only queries. Including them, will help us to validate our methodology. The high and low P&B values for the parameters used in this demonstration are given in Table V. All values are selected according to the recommendations made in PosgreSQL documentation [4]. For demonstration, we choose the high and low values for each parameter in a range such that it will act as monotonic.

V. Results

Table VI gives the SARD's estimated P&B effects and ranking of the configuration parameters for the queries of the workload consisting of TPC-H , {Q1,

Parameter	P&B Effects					Ranking				
	Q1	Q8	Q9	Q13	Q16	Q1	Q8	Q9	Q13	Q16
Checkpoint_timeout	63139	8910	602068	4130	1590	15	15	5	15	15
cpu_index_tuple_cost	10651	42106	149179	2460	1697	15	12	13	15	15
cpu_operator_cost	81071	46111	349389	480	1714	15	12	11	15	15
cpu_tuple_cost	58019	74791	48262	5808	1836	15	6	15	15	15
deadlock_timeout	11923	64500	54715	5214	1546	15	6	15	15	15
effective_cache_size	75846	11954	304263	238544	979	15	15	11	1	15
fsync	59809	4534	274194	3424	1329	15	15	11	15	15
geqo	3858	82480	529011	853	1679	15	6	7	15	15
maintenance_work_mem	75774	127031	780976	1351	1048	15	2	3	15	15
max_connections	97626	77876	628112	3678	1792	15	6	5	15	15
random_page_cost	8693	43699	1694532	3447	2145	15	12	1	15	15
shared_buffers	162386	34787	770430	132957	5997	15	12	3	2	2
stats_start_collector	71420	25181	328104	4278	2122	15	12	11	15	15
temp_buffers	101473	31217	571133	2713	1238	15	12	7	15	15
work_mem	5523703	359544	142035	24839	63760	1	1	13	3	1

TABLE VI. P&B effects of the configuration parameters for a workload consists of the TPC-H queries {Q1, Q8, Q9, Q13, Q16}.

Parameter	High Value	Low Value
effective_cache_size (pages)	81920	8192
maintenance_work_mem (pages)	8192	1024
shared_buffers (pages)	16384	1024
temp_buffers (pages)	8192	1024
work_mem (KB)	8192	1024
random_page_cost*	2.0	5.0
cpu_tuple_cost*	0.01	0.03
cpu_index_tuple_cost*	0.001	0.003
cpu_operator_cost*	0.0025	0.0075
Checkpoint_timeout (seconds)	1800	60
deadlock_timeout (milliseconds)	60000	100
max_connections	5	100
fsync	true	false
geqo	true	false
stats_start_collector	false	true

TABLE V. PostgreSQL configuration parameters and their P&B values used in the experimental setting. The symbol * indicates that the values are relative to the single sequential page fetch cost.

Rank	Parameter
1	work_mem
2	shared_buffers
3	maintenance_work_mem
4	max_connections
5	effective_cache_size
6	geqo
7	random_page_cost
8	Checkpoint_timeout
9	temp_buffers
10	cpu_tuple_cost
11	deadlock_timeout
12	cpu_operator_cost
13	stats_start_collector
14	cpu_index_tuple_cost
15	fsync

TABLE VII. The final ranking of the configuration parameters for a workload consists of the TPC-H queries {Q1, Q8, Q9, Q13, Q16}.

Q8, Q9, Q13, Q16}.The results indicates that for Q1, `work_mem` is the most important parameter; for Q8, `work_mem` and `maintenance_work_mem` are the most important parameters; for Q9 `random_page_cost`, `shared_buffers`, `maintenance_work_mem` are the most important parameters; for Q13, `effective_cache_size`, `shared_buffers` are most important parameters; and for Q16, `work_mem` and `shared_buffers` are important parameters. `work_mem` is ranked first in Q1, Q8, and Q9. `maintenance_work_mem` is ranked second and third respectively for Q8 and Q9. `shared_buffer` is ranked third, second, and second respectively in Q9, Q13, and Q16. `effective_cache_size` is ranked one in Q3.

In this paper, we assume that all queries have same importance in the workload. So intuitively it appears that `work_mem` is one of the most important configurations for tuning to improve the performance of the entire workload, as it appears as ranked first for three queries out of five. `shared_buffers` also looks very important for tuning as it ranked second for two queries

and third for another query. effective_cache_size and random_page_cost also looks promising as they ranked first for one query. Table VII listed the overall ranking of the all configuration parameters for the workload generated by SARD. From the result, it is interesting to note that overall ranking of the configuration parameters for the workload, are different from their rankings for the individual queries of the workload. For example, although random_page_cost ranked first for query Q9, but it does not appear in the topmost five important parameters for the workload considered.

Another interesting observation is that fsync and checkpoint_timeout never appear most important for the individual queries as well as the entire query workload. As the queries we considered here are read only, intuitively these two parameters should have marginal impact to improve the performance of the overall query workload. Experimental results match with intuition. The low ranking of these two parameters helps to validate that SARD is working correctly.

VI. Related Work

Recent research on database tuning can be classified into three broad categories: tuning the *physical design*, *identifying performance bottlenecks*, and *tuning the configuration parameters*. SARD falls into both the second and third categories.

Tuning the Physical Design: Major database vendors offer tools for automating database physical design. For example, Oracle 10g provides tools for selection of indexes, materialized views, for identifying the root causes of performance bottlenecks, and for estimating the benefit of eliminating a performance bottleneck [6], [14]–[16]. Microsoft SQL Server provides tools that allow for integrated selection of indexes, indexed views, and horizontal partitions [17]–[23]. IBM's DB2 recommends indexes, materialized views, shared nothing partitions, and multidimensional clustering of tables [2], [24]–[29].

Identifying Performance Bottlenecks: Rule-based decision trees are used to identify the potential sources of performance bottleneck [30]. A decision tree is formed based on a set of rules for finding the bottlenecks. ADDM uses a common performance metric 'database time' [6]. ADDM posses a holistic view of the database, identifies root causes of the performance bottlenecks, and estimates the benefits of eliminating performance bottlenecks. However, ADDM ignores system configuration parameters settings. A *design of experiments* based approach is used to evaluate the statistical significance of configurable parameters, the interaction effects between each parameter, web function types, and to rank key configurable system parameters that significantly impact overall system performance for E-Commerce systems [31], [32]. The drawback is that this an ad-hoc approach, lacking sound statistical methodology. Also, there is no upper bound on numbers of experiments needed to collect necessary data for determining rank.

Selecting Configuration Parameters: IBM DB2 UDB provides a wizard for automatically selecting the initial values for the configuration parameters [33]. Configuration choices are made by modeling each database configuration setting as a mathematical expression consisting of user specification of the database environments, automatically sensed system characteristics, and expert heuristics. Nonetheless, this feature suffers from the following problems: as the user specified inputs can have very different characteristics than the built-in workload models, the recommended values can be inaccurate; due to the use of heuristics, there is no statistical evidence that these values are correct; and no ranking is provided. In this work, we will address all these problems. As our conclusions are based on the real workloads, the first problem will not occur at all. The third problem is solved using a sound statistical methodology, which will indirectly take care of the second problem.

SARD uses P&B design for estimating the impact of a configuration parameter on DBMS performance for a particular workload. The P&B design has been used in various applications, such as improving the simulation methodology of micro architecture research [34]; identifying the most significant processor parameters, selecting a subset of benchmarks, analyzing the effect of an enhancement on processor performance [34]; characterizing and comparing existing computer architecture simulation techniques [35]; to systematically stressing statistical simulation by creating different performance bottlenecks [36]; and identifying the performance critical buses of a micro architecture [37]. To the best of our knowledge, SARD is the first to apply the statistical design of experiments based P&B design approach to study a DBMS.

VII. Conclusions and Future Work

SARD provides a methodology for ranking the configuration parameters based on their relative impact on DBMS performance. SARD is a generic methodology and can be applied to the non-database systems as well. A ranking based on statistical data will be a great aid for the DBAs to direct the tuning efforts. SARD can greatly serve this purpose. From our experience in this work, we

have come to the following two conclusions. First, when tuning individual queries, DBAs can use the ranking by rounding normalized effects method. Second, since the ranking of parameters for individual queries and the entire workload are different, DBAs must examine both to have a better idea of what parameters are important to consider for improving DBMS performance.

In the future, we are planning to perform the following extensions: 1) exploring more alternatives for ranking parameters for the individual queries as well as for the query workloads, 2) using SARD's estimated P&B effect to suggest the appropriate values of the configuration parameters, and 3) making SARD adaptable to the dynamic changes of the parameters and query workloads.

VIII. Acknowledgements

This work was supported in part by National Science Foundation grant no. CCF-0621462 and CCF-0541162, the University of Minnesota Digital Technology Center Intelligent Storage Consortium, and the Minnesota Supercomputing Institute.

References

[1] A. Rosenberg, "Improving Query Performance in Data Warehouses," http://www.tdwi.org/Publications/BIJournal/display.aspx?ID=7891, 2005.

[2] S. Lightstone, G. Lohman, P. Haas, V. Markl, J. Rao, A. Storm, M. Surendra, and D. Zilio, "Making DB2 Products Self-Managing: Strategies and Experiences," *IEEE Data Engineering Bulletin*, vol. 29, no. 3, pp. 16–23, 2006.

[3] C. Garry, "Who's Afraid of Self-Managing Databases?" http://www.eweek.com/article2/0,1895,1833662,00.asp, June 30, 2005.

[4] "PostgreSQL DBMS Documentation," http://www.postgresql.org/.

[5] S. Chaudhuri and G. Weikum, "Rethinking Database Architecture: Towards s Self-tuning RISC-style Database System," in *Proceedings of VLDB*, 2000, pp. 1–10.

[6] K. Dias, M. Ramacher, U. Shaft, V. Venkataramamani, and G. Wood, "Automatic Performance Diagnosis and Tuning in Oracle," in *Proceedings of CIDR*, 2005, pp. 1110–1121.

[7] G. Group, "The Total Cost of Ownership: The Impact of System Management Tools," 1996.

[8] H. Group, "Achieving Faster Time-to-Benefit and Reduced TCO with Oracle Certified Configurations," March, 2002.

[9] R. Plackett and J. Burman, "The Design of Optimum Multifactorial Experiments," in *Biometrika Vol. 33 No. 4*, 1946, pp. 305–325.

[10] "Transaction Processing Council," http://www.tpc.org/.

[11] D. J. Lilja, *Measuring Computer Performance A practioner's Guide*. Cambridge University Press, 2000.

[12] D. Montgomery, *Design and Analysis of Experiments*. Wiley, 2001.

[13] B. Debnath, J. Skarie, D. Lilja, and M. Mokbel, "SARD: A Statistical Approach for Ranking Database Tuning Parameters," *Laboratory for Advanced Research in Computing Technology and Compilers Technical Report*, no. ARCTiC 07-11, 2007.

[14] "Oracle Tuning Pack," http://www.oracle.com/technology/products/manageability/database/pdf/ds/tuning_pack_ds_10gr1.pdf.

[15] B. Dageville, D. Das, K. Dias, K. Yagoub, M. Zait, and M. Ziauddin, "Automatic SQL Tuning in Oracle 10g," in *Proceedings of VLDB*, 2004, pp. 1098–1109.

[16] B. Dageville and K. Dias, "Oracle's Self-Tuning Architecture and Solutions," *IEEE Data Engineering Bulletin*, vol. 29, no. 3, pp. 24–31, 2006.

[17] S. Agrawal, S. Chaudhuri, L. Kollar, A. Marathe, V. Narasayya, and M. Syamala, "Database Tuning Advisor for Microsoft SQL Server 2005." in *Proceedings of VLDB*, 2004, pp. 1110–1121.

[18] S. Agrawal, S. Chaudhuri, and V. Narasayya, "Automated selection of materialized views and indexes in sql databases," in *Proceedings of VLDB*, 2000, pp. 496–505.

[19] S. Agrawal, V. Narasayya, and B. Yang, "Integrating Vertical and Horizontal Partitioning into Automated Physical Database Design," in *Proceeding of SIGMOD*, 2004, pp. 359–370.

[20] N. Bruno and S. Chaudhuri, "Automatic Physical Database Tuning: A Relaxation-based Approach," in *Proceeding of SIGMOD*, 2005, pp. 227–238.

[21] ——, "To Tune or not to Tune?: A Lightweight Physical Design Alerter," in *Proceeding of VLDB*, 2006, pp. 499–510.

[22] S. Chaudhuri and V. Narasayya, "An Efficient Cost-Driven Index Selection Tool for Microsoft SQL Server," in *Proceeding of VLDB*, 1997, pp. 146–155.

[23] ——, "AutoAdmin What-If Index Analysis Utility," in *Proceeding of SIGMOD*, 1998, pp. 367–378.

[24] I. Ilyas, V. Markl, P. Haas, P. Brown, and A. Aboulnaga, "CORDS: Automatic Discovery of Correlations and Soft Functional Dependencies," in *Proceeding of SIGMOD*, 2004, pp. 647–658.

[25] S. Lightstone and B. Bhattacharjee, "Automated Design of Multidimensional Clustering Tables for Relational Databases," in *Proceeding of VLDB*, 2004, pp. 1170–1182.

[26] G. Lohman and S. Lightstone, "SMART: Making DB2 (More) Autonomic," in *Proceeding of VLDB*, 2002, pp. 877–879.

[27] M. Stillger, G. Lohman, V. Markl, and M. Kandil, "LEO - DB2's LEarning Optimizer," in *Proceeding of VLDB*, 2001, pp. 19–28.

[28] A. Storm, C. Garcia-Arellano, S. Lightstone, Y. Diao, and M. Surendra, "Adaptive self-tuning memory in DB2," in *Proceeding of VLDB*, 2006, pp. 1081–1092.

[29] D. Zilio, J. Rao, S. Lightstone, G. Lohman, A. Storm, C. Garcia-Arellano, and S. Fadden, "DB2 Design Advisor: Integrated Automated Physical Database Design," in *Proceedings of VLDB*, 2004, pp. 1087–1097.

[30] P. Martin, W. Powley, and D. Benoit, "Using Reflection to Introduce Self-Tuning Technology Into DBMSs," in *Proceedings of IDEAS*, 2004, pp. 429–438.

[31] M. Sopitkamol and D. A. Menascé, "A method for evaluating the impact of software configuration parameters on e-commerce sites." in *WOSP*, 2005, pp. 53–64.

[32] M. Sopitkamol, "Ranking Configuration Parameters in Multi-tiered E-Commerce Sites," *SIGMETRICS Performance Evaluation Review*, vol. 32, no. 3, pp. 24–33, 2004.

[33] E. Kwan, S. Lightstone, A. Storm, and L. Wu, "Automatic Configuration for IBM DB2 Universal Database," in *IBM Perfromance Technical Report*, January 2002.

[34] J. Yi, D. Lilja, and D. Hawkins, "A Statistically Rigorous Approach for Improving Simulation Techniques," in *Proceedings of HPCA*, 2003.

[35] J. Yi, S. Kodakara, R. Sendag, D. Lilja, and D. Hawkins, "Characterzing and Comparing Prevailing Simulation Techniques," in *Proceedings of HPCA*, 2005.

[36] A. Joshi, J. Yi, R. Bell, L. Eeckhout, L. John, and L. David, "Evaluating the Efficacy Statistical Simulation for Design Space Exploration," in *Proceedings of ISPASS*, 2006.

[37] V. Nookala, Y. Chen, D. Lilja, and S. Sapatnekar, "Microarchitecture-aware Floorplanning Using a Statisical Design of Experiments Approach," in *Proceedings of DAC*, 2005.