Developing a Characterization of Business Intelligence Workloads for Sizing New Database Systems

Ted J. Wasserman, Patrick Martin, and David B. Skillicorn
School of Computing, Queen's University
Kingston, Ontario Canada
1 (613) 533-6050
{tedjw, martin, skill} @ cs.queensu.ca

Haider Rizvi
IBM Toronto Laboratory
Markham, Ontario Canada
1 (905) 413-3160
haider@ca.ibm.com

ABSTRACT
Computer system sizing involves estimating the amount of hardware resources needed to support a new workload not yet deployed in a production environment. In order to determine the type and quantity of resources required, a methodology is required for describing the new workload. In this paper, we discuss the sizing process for database management systems and describe an analysis for characterizing business intelligence (BI) workloads, using the TPC-H benchmark as our workload basis. The characterization yields four general classes of queries, each with different characteristics. Our approach for sizing a BI application's database tier quantifies a new BI workload in terms of the response time goals and mix of the different query classes obtained from the characterization analysis.

Categories and Subject Descriptors
H.2.8 [**Database Management**]: Database Applications – *Data mining*

H.2.0 [**Database Management**]: General

General Terms
Design, Experimentation, Performance

Keywords
Workload Characterization, Clustering, Sizing, Business Intelligence, Capacity Planning

1. INTRODUCTION
Database systems continue to play a vital role in a company's IT infrastructure. Gartner [6] estimates that the total worldwide database software market in 2004 will reach $8815 million and $9203 million in 2005. In particular, specialized databases for data warehousing and business intelligence (BI) are becoming an important segment of the total database market.

BI workloads have very different characteristics than the traditional transaction processing workloads that most capacity planning and sizing methods are geared towards. These differences include [19]:

- A greater emphasis is placed on summarized and consolidated data versus a focus on individual records.
- A very large database size is used.
- Queries are heterogeneous, complex, and ad-hoc in nature and vary greatly in elapsed time.
- Queries often touch millions of records, and may perform many table joins, sorts, and aggregations.
- Queries sometimes produce very large results sets, requiring a lot of concurrent I/O.

While the performance tuning aspect of system configuration is vital, equally important is determining what type of hardware resources are required to support an application and its workload. This can be a complicated task because of the wide variety of processor, disk, network and memory technologies available. Further, determining the quantity of each resource needed and predicting how the different components interact with each other under a specific workload are not trivial tasks.

Computer system sizing attempts to arrive at a first estimate of a hardware configuration that will satisfy the performance demands, cost constraints, and functional requirements of an application. When sizing a system, we assume the following:

- Detailed information about the application and its workload is not available.
- Experts typically first find published performance results for a similar workload with similar performance requirements. They then extrapolate on these results using a combination of their experience, industry rules-of-thumb, and published performance relationships between different types of hardware. The result is a first estimate of the hardware configuration (processor, disk, and memory) needed to meet the resource demands of the expected workload and size of database.
- Customers expect a cost-efficient and effective hardware solution that meets the performance requirements of their application while offering the maneuverability to accommodate future expansion.

In this paper, we first discuss an approach for sizing database servers running BI workloads. We then describe a workload characterization analysis that we use as a basis for describing new BI workloads in our sizing approach. The analysis applies unsupervised data mining techniques to group individual BI queries into general classes of queries based on system resource usage. The resulting characterization provides insight into the resource demands of the queries typical of a BI workload and can be exploited in other areas such as workload management and meeting quality of service (QoS) requirements.

The remainder of the paper is organized as follows. Section 2 reviews related work. In Section 3, we highlight the main activities of our sizing process. Section 4 describes our workload characterization analysis. We summarize the paper and suggest some future work in Section 5.

2. RELATED WORK

Much of the research on sizing and capacity planning focuses on performance evaluation and workload prediction. Books such as Ferrari [4] emphasize tools and techniques for performance evaluation. Lam and Chan [14] examine the theory and approaches used in computer capacity planning in a more rigorous manner than previous studies. Industrial publications [8, 11] approach the topic from a more practical perspective and focus on specific hardware platforms. Previous work in our research group [2, 22] focuses on developing analytical models using Queueing Network Models (QNMs) to support capacity planning for DB2® Universal Database™ (UDB) [12]. Wasserman [19, 20] approaches sizing from an abstract perspective and determines the resources requirements of new workloads by extrapolating from the performance data of a known workload.

Workload characterization dates back to the 1970's when workloads largely consisted of large transactions and batch jobs that were run on mainframe computers. Techniques have evolved to accommodate some of the modern workloads encountered in today's computing environments. For a thorough review of the different characterization techniques in the literature, the reader is referred to Elnaffar and Martin [3]. Clustering is one of the most widely adopted techniques in workload characterization (for instance, see Calzarossa and Serazzi [1]; Nikolaou et al. [17]; Elms [5]).

Much of the recent clustering analysis work in the database and information retrieval fields is geared towards improving system performance. Ghosh et. al. [7] use clustering to reduce the complexity of selecting an optimal access plan for a query. In their approach, similar queries are grouped together into clusters and the optimizer-generated plan for the cluster representative is used to execute all future queries assigned to the cluster, resulting in significantly improved query optimization times. Wen et. al. [21] use clustering in the Information Retrieval domain to help human editors identify frequently asked questions. Their clustering strategy groups similar queries together more effectively than using keywords alone. Finally, Golfarelli and Saltarelli [9] use clustering to construct a profile of a data warehousing workload in order to summarize its characteristics. The profile can be used to help the designer during logical and physical optimization and to generate workloads useful for evaluating system performance in testing and benchmark settings.

Our analysis differs from previous work in two fundamental aspects, the choice of workload characterization features/parameters and the clustering algorithms used. The majority of prior analyses use a combination of structural query properties and statistical parameters to perform a clustering. The structural properties of a query are based on the text of the query itself, such as the number of table joins, the number of predicates, and the type of predicates. Statistical parameters are quantitative values contained in the database system catalog tables, such as the table size, the size and type of indexes on a table, and the skew of the data values in the tables. The parameters we use in our analysis are performance-oriented measurements. They are obtained by recording the query's consumption of system resources during query processing. The clustering algorithms we use also differ than those used in previous work. The techniques we employ are known for revealing underlying or hidden dimensions in data. This is particularly useful in our scenario because of the many different interactions that occur during query processing.

3. SIZING APPROACH OVERVIEW

It is important not to confuse the sizing process with capacity planning. While both processes have many activities in common, they have different objectives and methodologies. The majority of capacity planning studies and approaches in the literature assume that detailed performance measurements from a production environment are available to build models of system performance. Traditional capacity planning is also highly-geared towards estimating the future resource requirements of an *existing* workload. In contrast, sizing typically involves estimating the computing resources required to support a *new* workload that has not been run in a production environment. Since it is assumed that there are few production measurements available for the new workload, estimations, assumptions, and extrapolations must be made.

The capacity planning process also yields more accurate models of workload and system performance when performed correctly. This is attributed to the greater availability of detailed input. Conversely, a methodical sizing exercise has a much higher risk of inaccuracy than a capacity planning exercise, mainly due to the lack of factual data available about the new workload. There is also an inherent time-cost/accuracy trade-off between the two processes. The sizing process sacrifices some of the accuracy and deep understanding obtained through the capacity planning process in return for a quick and straightforward study.

Our sizing approach entails two main activities: estimating the performance requirements of the new workload and selecting hardware configurations that meet these requirements. The overall process can be summarized as follows:

1. Collect the required high-level input data from the customer.
2. Cross-check and verify input data, making assumptions and estimates if needed.
3. Determine the required system resource demands for each workload class and type.

4. Aggregate the different workload class and type resource demands to determine the overall system requirements.
5. Determine which hardware configurations meet the required resource demands.
6. Produce a ranked list of hardware configurations according to desired criteria and recommend a configuration to the customer.

While these steps are general to many relational database management systems (DBMSs) and workloads, the implementation details are workload and/or DBMS-specific.

In our approach, the resource requirements of *new* online/production workloads are calculated by extrapolating from the observed performance of queries from the Transaction Processing Performance Council (TPC) TPC-H™ benchmark [18] as well as using various assumptions, projections, and rules-of-thumb. The requirements of batch/ETL workloads are calculated using a series of equations that take into account parameters such as the input volume, transform complexity, and time window. An algorithm, based on our chosen units of processing power, is used to determine the minimum processor configuration required for each desired server model. Memory is determined for each server configuration using rules-of-thumb. Networking requirements are not considered in the approach at this time and are suggested as future work. An algorithm for sizing disk subsystems takes into account the peak throughput requirement of the workload and uses pre-configured disk systems with known throughput rates to determine how many of such units are required. The amount of raw storage required is determined using rules-of-thumb. For a full description of the sizing process, including calculations and algorithms, the reader is referred to Wasserman [19, 20].

In order to perform Step 3 in our sizing approach, a straightforward technique is needed to describe a new online/production workload. We propose one such technique for doing so in the next section.

4. CHARACTERIZING BI WORKLOADS

BI applications and their workloads vary depending on the type of application, the target industry, and the nature of business questions being answered. We use the TPC-H benchmark as our representative BI workload for this task. The TPC-H benchmark consists of 22 ad-hoc queries which answer questions representative of any industry which must manage, sell, or distribute a product worldwide (car rental, food distribution, parts, suppliers, etc.).

It is difficult to create an accurate workload model if we view a BI workload as a single collection of homogeneous queries. A model also becomes too complex if we treat each query individually, in effect, as its own workload. A more practical solution is to partition queries into a few general classes based on their resource usage. Each class will comprise the queries that are similar to each other based on resource usage and other relevant characteristics.

For our analysis, we follow the general methodology described by Menascé et. al. [16] to construct a workload model. It is based on a resource-oriented characterization of workloads and includes the following steps:

1. Identification of the basic components.
2. Choosing characterizing parameters.
3. Data collection and normalization.
4. Partitioning the workload into classes.
5. Identifying the class characteristics.

We now describe these steps in detail as it applies to our chosen workload.

1. Identifying workload components

We use the 22 queries of the TPC-H benchmark as our representative BI workload. TPC-H was designed to simulate the type of resource activity commonly found in BI applications. TPC-H can be run in different modes of execution. We are interested in the "power run" mode of execution where a single-stream of queries is submitted to the system, with each query being executed one after another in a serial fashion.

2. Choosing characterizing parameters

For our analysis, we monitor several performance-oriented parameters for each query, including:

- Response time - the amount of time elapsed, in seconds, from when a query is submitted to the system to the time the result set is returned.
- Average processor (CPU) utilization - the average utilization of the processor(s) over the duration of query execution. Note that this only includes the utilization of user processes, as opposed to the operating system kernel or privileged threads/processes.
- Sequential Input/Output (I/O) throughput rate - the average rate that data is read from disk over the duration of query execution, measured in megabytes per second (MB/second).
- Random I/O operations - the average rate of I/O instructions processed over the duration of query execution, measured in I/O operations per second (IOPS).

We are also able to monitor other types of parameters, such as memory and network utilization; however, the above set is sufficient for our analysis.

3. Data collection and normalization

The input data for our analysis is obtained from six recent pre-audited DB2 UDB TPC-H benchmark runs. Since the data we use was not audited, we can only claim that it is "TPC-H like". From now on, when we refer to our TPC-H data, we really mean to say "TPC-H like" data. Benchmarks were conducted by the

IBM DB2 UDB Performance Group and used IBM DB2 UDB version 8 as the DBMS. The computer models, hardware parts, operating systems, and database scale varied across benchmarks. Care was taken to ensure that each benchmarked system configuration was *balanced*. A system is considered balanced when all of its resources are working in concert to allow the optimal amount of workload through the system to meet specific objectives. In the analysis, we assume that the performance of a balanced system is relative to the amount of system resources available. That is, if we increase the quantity of system resources in a balanced fashion, system performance will increase. Although one benchmark configuration might be more powerful than another, we expect this will be reflected in improved performance.

Raw data was monitored and collected using standard operating system performance monitoring tools, which were configured to sample the desired parameters at five second intervals. The final parameter values for each query were determined by averaging all the raw data samples collected over the interval, for each respective parameter.

The choice of units of measurement can affect the characterization analysis. For instance, expressing temporal data in seconds versus hours could lead to a very different result, depending on what type of analysis technique is used. To help avoid dependence on the choice of units, the data is standardized. In standardizing the data, we attempt to give all parameters an equal weight. This is particularly important in our case, since the values we measure for each query are partially influenced by the configuration of system they are run on, as well as on other factors such as database scale. To normalize the data, we calculate the z-score of each measured parameter variable. Z-scores transform the dataset into one with a mean of 0 and standard deviation of 1. The z-score of a parameter value can be calculated as shown in Equation 1:

$$z\ score = \frac{measured\ value - mean\ value}{standard\ deviation} \qquad (1)$$

Once each benchmark's data is normalized, we can combine all the benchmark data into a single matrix (table) for use in the next stage of the characterization process, workload partitioning.

4. Partitioning the workload into classes

We use clustering techniques to partition our workload. Clustering is the process of grouping data into classes or clusters so that objects within a cluster are similar to each other, but are dissimilar to objects in other clusters. The techniques we use are a combination of Singular Value Decomposition (SVD) [10] and SemiDiscrete Decomposition (SDD) [13]. Both techniques are examples of unsupervised data mining, where the goal is to discover structured information in the dataset without knowing or providing hints as to what that structure might look like. They are also examples of matrix techniques, which view the dataset as a matrix and attempt to decompose the matrix in various ways depending on the goal of the analysis. SVD and SDD both decompose a dataset matrix as the product of three new matrices. However, the meaning and structure of each matrix is different.

A SVD decomposes a dataset matrix A into the product of three matrices, U, S, and V such that:

$$A = U\,S\,V^T$$

where U is n x m, S is a diagonal matrix of non-increasing non-negative values, and V is m x m. In effect, SVD transforms an m-dimensional space into a new m-dimensional space such that the new axes are orthonormal and the axes are ordered so that the maximum amount of variation is contained in earlier axes. The entries of S are scaling factors indicating the relative importance of each axis. Geometrically, the rows of U represent coordinates of the corresponding rows of A in a space spanned by the columns of V, while the rows of V represent the coordinates of the corresponding columns of A. A common practice in SVD is to truncate the representation to k dimensions, where k is some arbitrary constant, to make analysis more manageable. Since SVD concentrates as much variation as possible into the earliest dimensions, truncating is feasible because the least possible information is discarded.

A SDD is similar to a SVD in that it decomposes a dataset matrix A into three matrices, such that:

$$A = X\,D\,Y$$

However, the components of the decomposition have a different form and meaning than SVD. X is an n x k matrix, D is a k x k diagonal matrix, and Y is a k x m matrix, where k is an arbitrary constant. The entries of X and Y are from the set {-1, 0, +1}. Objects are divided based on their value in the first column of X {-1, 0, +1}. They can be subdivided again according to their values in the subsequent columns of X. In effect, SDD discovers rectilinearly aligned regions of the matrix of similar (positive and negative) magnitude. These regions/partitions determine which objects are related to each other.

SVD and SDD can be jointly applied to our dataset by using both decompositions, truncating the SVD at $k = 3$, plotting the points corresponding to queries, and labeling each point according to its location in the top few levels of the SDD decomposition.

The analysis of our TPC-H benchmark data, including matrix calculations and plots, is performed using MATLAB [15]. We include one additional attribute in the analysis, the size of the largest n-way table join for each query. The addition of this extra attribute results in a tighter clustering of data points, due to the fact that queries with the same label are more closely related. The results from analyzing the query dataset are shown in the following figures. Here, the position of the points is determined by SVD, while the shape and color used to distinguish among them is determined by SDD.

Before we begin our analysis, we first examine the relevancy of the performance attributes we measured for the queries. Figure 1 shows the result of a joint SVD and SDD classification on the attributes. This plot illustrates the relative contribution of each attribute to the analysis. Since the five points corresponding to the five attributes used are equally spread out in geometric space, we can conclude that each attribute is significant and adds useful information to the analysis.

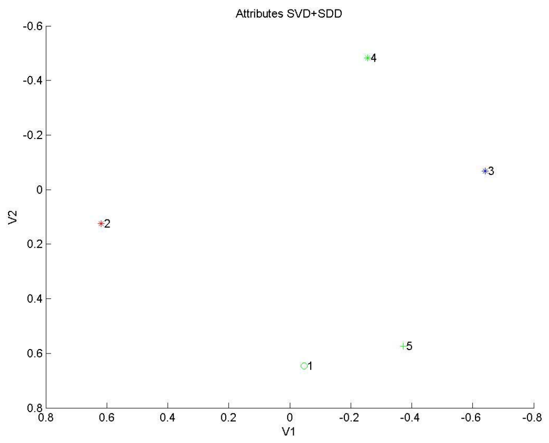

Figure 1. SVD + SDD plot of attributes, dimensions 1 and 2

Figure 2 shows the results of a joint SVD and SDD classification. There are a couple of important pieces of information we can extract from this plot:

- Four clusters of queries appear to be present in this dataset. We have labeled each cluster in the figure as well as provided approximate cluster boundaries (illustrated using dotted lines). The cluster boundaries are determined by analyzing the raw data values produced in the analysis by MATLAB (not shown). Visual inspection of the different plots of the data is also helpful for determining the approximate location of the cluster boundaries. Based on our analysis, the queries belonging to each cluster are as follows:

  Cluster 1: Q1, Q3, Q4, Q5, Q6,

  Q11, Q14, Q14, Q19

  Cluster 2: Q2, Q20

  Cluster 3: Q7, Q8, Q9, Q18, Q21

  Cluster 4: Q10, Q13, Q15, Q16, Q22

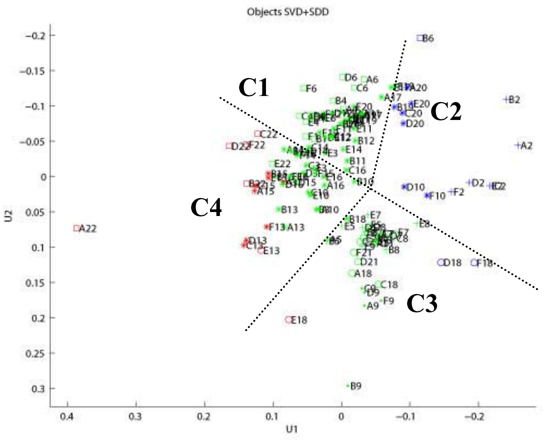

Figure 2. SVD + SDD plot of queries, dimensions 1 and 2

- Queries appear to scale well across different system architectures and benchmark scales. For instance, all the query 1 points appear very closely together in the plot. The same is true for most of the other queries.

5. Identifying class characteristics

At a high level, we develop an understanding of what each cluster represents. Cluster 2 appears to represent fairly simple complexity queries that are generally IO-bound in nature and have a small number of tables being joined. Cluster 3 represents long-running, large and complex queries with a large number of tables being joined (5+). Queries in this group also exhibit high sequential and random IO usage. Cluster 4 represents short-running, trivial complexity queries with a varying amount of tables being joined (3-8). Finally, Cluster 1 appears to represent moderate-complexity queries with a smaller number of tables being joined (1-5) and exhibiting high CPU utilization. We should also note that Cluster 1 is considered less interesting than the others since the data points in that cluster are those closest to the origin in the plot.

To lend support to our belief about the meaning of the different clusters, we generate data points for artificial queries that we believe belong in each of the clusters and show that they do in-deed fall into the cluster, hopefully at the cluster center. Figure 3 is similar to Figure 2 except for the addition of four artificial points, *X1*, *X2*, *X3*, and *X4*, created to confirm our understanding of the different clusters. As we hoped, each new point is in close proximity to the center of its intended cluster, lending support to our belief about the cluster semantics.

In general, there is no practical way to decide what the clusters represent, since the SVD transforms the original axes into a new set of pseudo-axes whose meaning is not immediately clear. One way to make a stronger argument about the validity and appropriateness of the clustering is to try and understand the meaning of the new dimensions. This is again accomplished by creating artificial query data points; however, this time, we try to generate points that represent extreme examples of what each new dimension is thought to represent. If the SVD transformation of these additional points result in them being placed at the extreme ends of one of the transformed dimensions, we can be more confident about our understanding of the new dimensions and clustering. Figure 4 is a plot similar to Figure 2 except for the addition of two artificial points, *AU1a* and *AU1z*, created to confirm what we believe is causing the variance in the *U1* dimension. Both points appear at the extreme ends of dimension *U1*, meaning we have a fairly good understanding of what the dimension represents: queries that are CPU-bound versus those that are IO-bound.

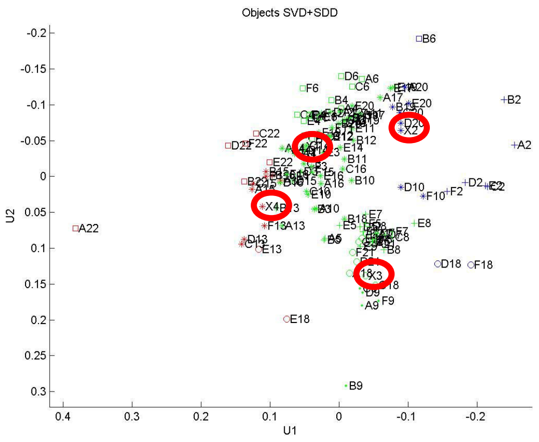

Figure 3. SVD + SDD plot of queries, dimensions 1 and 2 with four artificial points targeting cluster centers

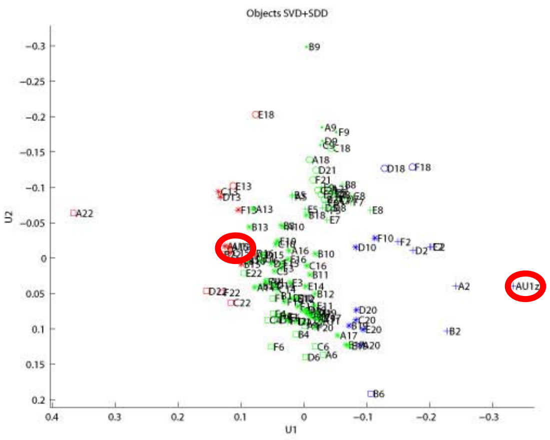

Figure 4. SVD + SDD plot of queries, dimensions 1 and 2 with two additional points representing extremes of dimension 1.

A similar analysis is repeated for dimensions *U2* and *U3*. Figure 5 and Figure 6 show plots similar to Figure 2 except for the addition of two artificial points for what we believe will reveal the meaning of the *U2* and *U3* dimensions, respectively. Each set of points appear at the extreme ends of their respective dimensions in both plots. Based on this evidence, we believe that the variance along dimension *U2* is caused by differences in query response times, while the variance along dimension *U3* is thought to distinguish between queries that are sequential-I/O intensive and random-I/O intensive. We should note that we can not be as conclusive about the meaning of dimension *U3* since our artificial points do not fall exactly at the extremes as we had hoped. There are perhaps other contributing hidden factors we have not considered. Since dimension *U3* is not as significant as the first two, further analysis is left as future work.

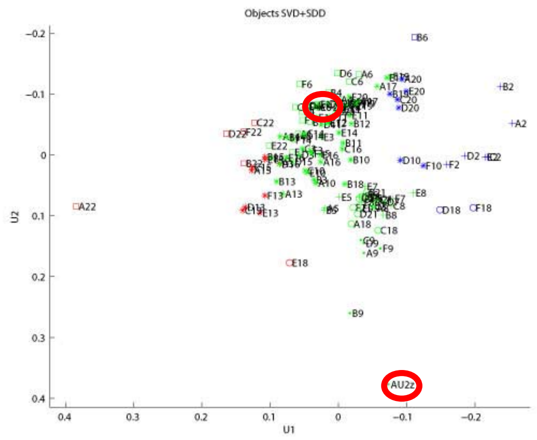

Figure 5. SVD + SDD plot of queries, dimensions 1 and 2 with two additional points representing extremes of dimension 2.

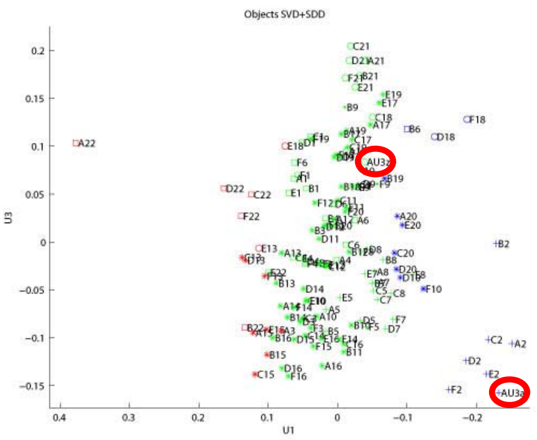

Figure 6. SVD + SDD plot of queries, dimensions 1 and 3 with two additional points representing extremes of dimension 3.

5. CONCLUSION AND FUTURE WORK

The work presented in this paper is a good step towards a robust approach for sizing the database tier of a BI application. The main contributions of this paper are an overview of the database sizing process and a characterization analysis of BI workloads.

The characterization analysis was needed in order simplify the task of describing new workloads during the sizing process. A workload characterization based on resource demands can also be useful for other important administrative tasks such as workload management and handling QoS requirements.

The analysis used a resource-oriented characterization of the 22 queries of the TPC-H benchmark. Performance data was collected from six non-audited "power runs" of different TPC-H benchmarks. The data collected from each system was first normalized to remove effects of unit and scale, then combined into a single dataset for use in a joint SVD and SDD classification analysis. From the analysis, it was shown that the queries of TPC-H can be grouped into four broad categories, each with different characteristics. One group describes trivial complexity queries, with short run times, a small number of tables being joined, and exhibiting high CPU utilization. Another group represents simple complexity queries which are I/O-bound and have a small number of tables being joined. Another group represents medium complexity queries with moderately high response times and moderate CPU and I/O usage. Finally, one cluster represents large and complex queries which are long-running, have a large number of tables being joined, and exhibit high sequential and random I/O usage.

We demonstrate that our understanding of the different query clusters is correct by first generating performance data for artificial queries that we believe belong in each cluster, then showing that they do. Second, we attempt to understand the meaning of the new dimensions in the SVD space by considering the what causes the variance in each dimension.

Our characterization of a BI workload in terms of trivial, small, medium and large complexity queries is used in our sizing approach for describing new workloads. The workload of a new BI application is described in terms of the quantity and response time goals of these different query classes.

Future work in this area includes improving the quality and robustness of the analysis by including performance data from a greater number of benchmarks as well as from real customer production environments. Extending the analysis to different workload types such as OLTP would also be beneficial. In this case, characterization and sizing will need to be done differently since these workloads exhibit different resource consumption patterns and are affected by different types of inputs.

6. ACKNOWLEDGEMENTS

This research is supported by IBM Canada Limited, the Government of Ontario, and Communication and Information Technology Ontario (CITO).

7. TRADEMARKS

IBM, DB2, and DB2 Universal Database are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

8. REFERENCES

[1] M. Calzarossa and G. Serazzi. "Construction and Use of Multiclass Workload Models", Performance Evaluation, 9, 4, 1994.

[2] X. Cui, P. Martin, and W. Powley. "A Study of Capacity Planning for Database Management Systems With OLAP Workloads", Proceedings of CMG 2003, December 2003.

[3] S. Elnaffar and P. Martin. "Characterizing Computer Systems' Workloads", Technical Report 2002-461, School of Computing, Queen's University, Canada, 2002.

[4] D. Ferrari, Computer Systems Performance Evaluation, Prentice Hall, 1978.

[5] C. Elms. "Clustering - One method for Workload Characterization", Proceedings of the International Conference on Computer Capacity Management, San Francisco, California, 1980.

[6] Gartner, Press Room - Quick Statistics. Retrieved from http://www.dataquest.com/press_gartner/quickstats/databases.html on June 1, 2004.

[7] A. Ghosh, J. Parikh, V.S. Sengar and J. R. Haritsa. "Plan Selection Based on Query Clustering", Proceedings of VLDB, 2002.

[8] G.B. Gibbs et. al. IBM e-server pSeries Sizing and Capacity Planning: A Practical Guide. IBM Redbook SG-247071, March 2004, http://www.redbooks.ibm.com/redbooks/pdfs/sg247071.pdf.

[9] M. Golfarelli and E. Saltarelli. "The Workload You Have, the Workload You Would Lik", Proceedings 6th ACM International Workshop on Data Warehousing and OLAP (DOLAP 2003), 2003.

[10] G.H. Golub and C.F. van Loan. Matrix Computations, Johns Hopkins University Press, 3rd edition, 1996.

[11] S. Hahn et. al. Capacity Planning for Business Intelligence Applications: Approaches and Methodologies, IBM Redbook SG24-5685, November 2000, http://www.redbooks.ibm.com/redbooks/pdfs/sg245689.pdf

[12] IBM. DB2 Universal Database, http://www.software.ibm.com/data/db2/udb.

[13] T.G. Kolda and D.P. O'Leary. "Computation and uses of the semidiscrete matrix decomposition", ACM Transactions on Information Processing, 1999.

[14] S. Lam and K. H. Chan. Computer Capacity Planning: Theory and Practice, Academic Press, 1987.

[15] Mathworks Incorporated. MATLAB, http://www.mathworks.com.

[16] D. Menascé, V.A.F. Almeida, and L. Dowdy. Capacity Planning and Performance Modeling: From Mainframes to Client-Server Systems, Prentice Hall, 1994.

[17] C. Nikolaou et. al. "The Impact of Workload Clustering on Transaction Routing", Technical Report FORTH-ICS TR-2238, 1998,

[18] Transaction Processing Performance Council. TPC Benchmark H Standard Specification, Revision 2.1.0 http://www.tpc.org/tpch/spec/tpch2.1.0.pdf.

[19] T.J. Wasserman. Sizing Database Systems for Business Intelligence Workloads, M.Sc. Thesis, School of Computing, Queen's University, 2004.

[20] T.J. Wasserman, P. Martin, and H. Rizvi. "Sizing DB2 UDB Servers for Business Intelligence Workloads", Proceedings of CASCON 2004, October 2004.

[21] J-R Wen, J-Y Nie and H-J Zhang. "Query clustering using user logs", ACM Transactions on Information Systems, 20, 1, 2002.

[22] H. Zawawy, P. Martin and H. Hassanein. "Supporting Capacity Planning for DB2 UDB", Proceedings of CASCON 2003, September 2003.