International Journal of Physical Sciences Vol. 7(9), pp. 1482 - 1492, 23 February, 2012
Available online at http://www.academicjournals.org/IJPS
DOI: 10.5897/IJPS11.081
ISSN 1992 - 1950 ©2011 Academic Journals

Full Length Research Paper

Workload management: A technology perspective with respect to self characteristics

Abdul Mateen[1]*, Basit Raza[1], Muhammad Sher[1], M. M. Awais[2] and Norwatti Mustapha[3]

[1]Department of Computer Science, International Islamic University, Pakistan.
[2]Department of Computer Science, Lahore University of Management Science, Pakistan.
[3]Faculty of Computer Science and IT, University Putra Malaysia, Malaysia.

Accepted 19 October, 2011

Rapid growth in data, maximum functionality and changing behavior tends the workload to be more complex. Organizations have complex type of workloads that is very difficult to manage by the humans and even in some cases, this management becomes impossible. Human experts take much time to get sufficient experience so that they can manage workload efficiently. The versatility in workload due to huge data size and requests (workload) lead us towards new challenges. One of the challenges is the identification of the problems queries and the decision about these, that is, whether to continue their execution or stop. The other challenge is how to characterize the workload, as good configuration, prediction and adoption is fully dependent on characterization of the workload. Correct and timely characterization leads to managing the workload in an efficient manner and vice versa. In this scenario, our objective is to produce such workload management strategy or framework that is fully autonomic. This paper provides some basis and achievements about the tools that exhibit autonomic computing (AC) in workload management with respect to self-characteristics. We have categorized the workload tools to these self-characteristics and identified their limitations. Finally the paper presents the research done in workload management tools with respect to workload type and autonomic computing.

Key words: Autonomic computing, workload, optimization, configuration, prediction, organization, adoption.

INTRODUCTION

The systems, which can execute, adjust and tune themselves in the presence of workload, are called autonomic computing (AC) systems. The theme of AC systems is to focus on what rather than how. The term autonomic computing was first time introduced by the IBM in 2001 to describe the systems that can manage themselves without any human interaction. The inspiration of the AC is taken from the human nervous system that performs different activities without conscious thought. An AC system would control the functioning of computer applications and systems without or minimal human intervention, in the same way human autonomic nervous system regulates body system without conscious input from the individual. The basic purpose of the AC is to create such systems that have the ability to run themselves with hiding the complexity from the user (Horn, 2001; Parashar and Hariri, 2005). The concept of self-management and adoption in computing system is very old. In the past, most processes are automatic but afterward it has been realized that the processes should be autonomic (Huebscher and McCann, 2008; Beg et al., 2010; Ejaz and Baik, 2011).

Automatic means pre-programmed task execution of a system, that is, system remains in working until something goes wrong and human intervention is necessary for further execution. While autonomic means self-regulation, here system response is also automatic, but modulated and system can handle problems itself with no human intervention (Mateen et al., 2008). AC is an evolutionary process rather than revolutionary process. The implications for computing are a network of

*Corresponding author. E-mail: abdulmateen@fuuastisb.edu.pk Tel: (+92)3335275393

organized computing components that give us what we need, when we need it, without a conscious mental/physical effort. AC is a self-managing computing model. The basic purpose of AC is to create such systems that have the capability to run themselves and hiding complexity from users.

AUTONOMIC WORKLOAD MANAGEMENT

In workload management, the main functions are workload frequency patterns, composition, intensity and required resources. The complexity in DBMSs increases due to the various functionality demands from the users, complex data types, diverse workload and data volume are increasing with the passage of time. All these factors cause brittleness and unmanageability in DBMS. To handle this problem, organizations hire number of expert database administrators (DBAs) and spending lot of money to get expected improvement, throughput and response. As shown in Figure 1, usually DBA have to take care of all the tasks such as making policy for workload priorities, memory, configuration and other database management system (DBMS) related tasks. The cost of hardware is decreasing but the cost of management is increasing. Performing workload management activities manually, by hiring experts causes increase in total cost of ownership (TCO). Moreover with the advent of distributed systems and data ware house, it is become difficult and even some cases impossible for DBA to manually organize, optimize and configure day to day tasks. To achieve better workload management, executing queries may be stopped for a while and later these can be restarted. However, when queries will be stopped then the executed work will be lost even that may be about to complete and will be executed from the scratch if essential.

In workload management, there are three units that are workload, resources and objectives that are co-related with each other. The workload uses some resources to meet the objectives of an organization or we can say resources are allocated through different approaches to workload which has some management objective. Workload has evolved through three phases which are capacity planning, resource sharing and performance oriented workload and the style has been changed from offline to online. In capacity planning workload management, the main purpose was cost sharing; in resource oriented the idea was maximum resource utilization while in case of performance oriented, the focus is on the business objectives. The style of workload has been changed from the offline analysis to online adaptation. The adaptation of workload consists of workload detection and workload control. Workload is detected through two methods, one is workload characterization and other is performance monitoring. When performance degrades, the characterization

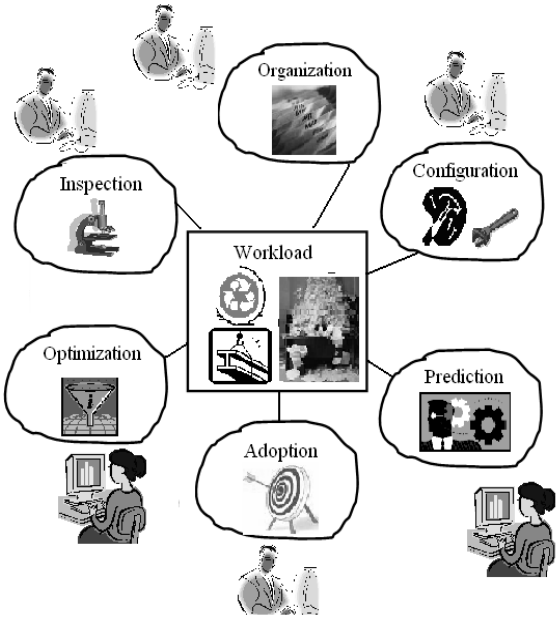

Figure 1. Workload management activities.

method is used by tracking the workload changes proactively while the performance monitoring method is reactive, which take action when performance has been degraded. Three techniques are used to derive workload control plan, which are performance model, heuristic and threshold technique.

The unmanageability in workload management can be handled by making the DBMS to self-manage (Autonomic) that can perform its tasks such as memory, configuration, storage management and resource allocation automatically according to the current environment. The benefit of autonomic computing is to manage complexity itself according to the set goals and objectives. Autonomic computing contributes an important role in managing systems, database management systems and workload management. Workload management is a main feature in DBMS and should be autonomic to improve the efficiency of DBMS. Different techniques, models and tools have been developed by different vendors and practitioners to handle workload autonomically. These tools are about workload scheduling, multiprogramming, prediction, adoption and resource allocation. The technology of autonomic computing in workload management is used to mange the workload in an efficient and responsive way. There is need of workload manager that manage workload without affecting other requests, efficient resources utilization and handle all other matter related with workload. The autonomic workload manager will perform these tasks by collecting information about workload type, intensity, resource demand etc with minimal human intervention. This autonomic technology has a high potential to be incorporated in current DBMSs.

SELF-MANAGEMENT IN AWM

Autonomic workload management should have self-optimization, self-configuration, self-inspection, self-prediction, self-organization and self-adoption features. Self-optimization in workload management exhibits that all task related with workload must be executed in an efficient manner. In order to achieve efficiency in workload management, configuration of different components should be performed in self-manage and appropriate way. Self-inspection in autonomic workload management supports better decisions making by using the knowledge of its resources, limits, intensity etc. Self-prediction in workload management helps to forecast the different aspects such as resource demand, workload frequency and memory requirements etc for the future. Self-organization in autonomic workload management allows reorganizing and restructuring the layout of data and indexes in order to make improvements. Self-adoption allows adopting the changes in workload according to the available resources and environment. Autonomic workload management has the following characteristics:

Self-optimization

Self-optimization is the characteristic that is responsible to execute the task or utility in an efficient manner. In case of DB workload, self-optimization is the way to execute the DB workload in an efficient and organized way according to the available resources and environment. Much of the research has been done in the context of workload management with respect to self-optimization. In the next paragraph, available workload optimization techniques in DBMSs and DWs would be discussed.

Oracle database resource manager (ODRM) (Rhee et al., 2001) allows the DBA to logically divide the workload into distinct units and allocates CPU resources to these units without extra overhead. During peak hours OLTP workload should be given preference over DSS queries and vice versa. ODRM has scheduling mechanism that is used for fixed time interval and controls the number of active sessions executing at a time. When the available slots for active sessions are filled with new sessions, the remaining sessions will be queued until some slot is available. Through ODRM, administrator can define and set the scheduling policies for workload based on the predicted execution time of a query. The major components of ODRM are Resource Consumer Group, Resource Plan and Resource Plan directive.

The research (Mumtaz et al., 2003, 2008, 2009) discussed the impact of query interaction over the workload and introduced a framework named as Query Shuffler (QShuffler) and shown in Figure 2. The proposed framework schedules the workload by considering the impact of queries. Requests are given by the users in the form of queries; the QShuffler classifies these queries according to their type and arranges them in an efficient way. The QShuffler adopts the non-preemptive scheduling technique and minimize the dropping requests using shortest remaining time first scheduling technique. It is evaluated with TPC-H workload in DB2.

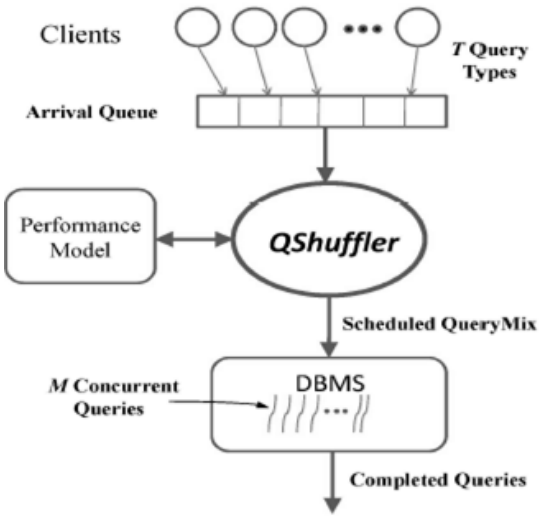

Figure 2. Query Shuffler [Mumtaz et al., 2009].

BI Batch manager (Mehta et al., 2008)) is introduced for enterprise data ware house that take queries in the form of batches. It consists of three components which are admission control, scheduler and execution control. Admission control works on the basis of memory requirement in the form of batches. A batch consists of those queries whose memory requirement is equal to the available memory of the system. For scheduling, the authors proposed that the query with maximum memory requirement will have the highest priority. The manager uses Priority Gradient Multiprogramming (PGM) to prevent under load and over load problems. Finally some experimental results are discussed to evaluate performance of BI batch manager.

In Oracle, the automatic SQL tuning is performed through query optimizer and SQL tuning advisor (Dageville et al., 2004). Query optimization has great importance especially in case of complex workload. The SQL Tuning Advisor is an interface between optimizer and user. It generates tuning recommendations for SQL statements (workload). These recommendations are provided to user, who either select or reject. When user selects recommendations, it will be stored in SQL profile that is further utilized by Oracle query optimizer for generation of best query execution plans. SQL Tuning Advisor makes different decisions on basis of information that is provided by the query optimizer, Automatic Database Diagnostic Monitor (ADDM) and Automatic Workload Repository (AWR).

Mehta et al. (2009) define the design criteria that make a Mixed Workload Scheduler (MWS) and use it to design rFEED, that is, MWS that is fair, effective, efficient, and differentiated. They proposed a non-preemptive approach for scheduling as for them it is expensive to preempt small queries that make the bulk of a BI workload. The approach uses optimizer's estimated execution cost as an approximation as authors thought that approximation is sufficient and no need to use precise value. Moreover, a single queue for scheduler and multiple queues for execution are used. They also assumed that all queries have same normalized service level. The authors simulated real workloads and compare it with models of the current best of breed commercial systems.

Surajit et al. (2007) proposed a framework to stop and restart the queries during their execution to manage the workload efficiently. The restart approach re-executes the stopped query from the position where it was stopped. This technique does not save all the executed information but save only the selected information from the past execution to reduce memory overhead. This method also reduces the running time of re-executed queries. The proposed technique is validated by making experiment over real and benchmark data (TPC-H).

A technique for query suspension and resumption (Chandramouli et al., 2007) with minimum overhead is discussed where the author proposed induction of asynchronous checkpoints for each cardinality in a query. Authors proposed an optimized plan for suspension which dumps the current state to disc and going back to previous checkpoint. The optimized plan performs its tasks (suspension or resumption) with less overhead by observing the time constraint during suspension. The proposed approach is implemented in PREDATOR tool. The technique in Query Suspension and Resumption has proven experimentally for simple and heavy workload and it is observed that it meets suspend time constraint and thereby reducing the overhead. The technique uses hybrid approach for query suspension where suspend time overheads are negligible and due to this, better results can be seen. The memory wastage is higher in previous techniques due to switching points and shows worse results for unexpected suspend.

Schroeder et al. (2006) proposed external scheduling technique for OLTP workload. To select the appropriate MPL, they identified main parameters and used feedback controller to select the MPL automatically that is based on queuing theoretic model and Markov chain. Priorities are used for external scheduling. The technique is validated through experiments and observed that external scheduling can be equally effective to internal scheduling when suitable MPL is selected.

QShuffler (Mumtaz et al., 2003, 2008, 2009) considers the impact of queries over each other as it has vital role in performance and proposed an experimental technique to handle query interaction problem. QShuffler gives optimal solution for scheduling of workload as it is based on linear programming. It gives four times performance over the FCFS scheduler of database systems. However in QShuffler, the large jobs have to wait for a long time even these are of higher priority as it uses SJF algorithm for scheduling. The average execution time is larger as QShuffler uses non-preemptive SJF approach. Moreover this approach can be improved if the service level objectives (SLOs) are incorporated with scheduling algorithm. BI Batch Manager (Mehta et al., 2008) executes the workload in the form of batches to avoid thrashing and provides the optimal solution for all types of workload. This approach do not require any changes in the internals of DBMS to manage small and heavy workload .The manager gets benefit of the added predictability of queries; stabilize the system and less sensitive to estimation errors. It uses feedforward loop that stabilizes the system (from underload or overload) with maximum ability to tolerant the prediction errors. In Feedback control, technique samples the performance metric and controls the incoming request optimally. When performance metric exceeds from the threshold, the rate of admitting requests reduces and vice versa. It gives high throughput for commercial and enterprise class DBMS; however it has no ability to handle interactive and ad-hoc queries. Dynamic approach should be adopted as the workload varies from time to time. The sub-batches are created on the basis of memory only. As the query with largest memory gets higher priority so the queries with small memory will wait for a long time. Due to this reason, throughput decreases and starvation occurred. As the PGM executes the first query which requires the largest memory, the same problem seen with LMP will also occur in PGM. The suggested methodology in Mehta et al. (2009) uses non-preemptive scheduling scheme that gives very poor results for time critical systems due to its poor responsiveness and ultimately there is a chance of starvation and hanging. Authors used the approximate values that never give the actual results. Optimizer's estimated execution cost is used but in real life approximate values never give the actual results. Single queue is maintained for scheduling, so global optimization cannot be achieved. The proposed methodology assumes the same service level but in real life, the workload or queries do not have the same service level. The approach set the MPL statically and does not consider the interaction among different queries. SQL Tuning Advisor (Dageville et al., 2004) improves the execution plans through SQL Profiling concept; and on the basis of cost-based access paths and what-if analysis, it generates tuning recommendations for the incoming workload. It has a very strong analysis capability where it performs a number of analyses such as estimate, access, parameter setting, statistical and SQL structure analysis. Whenever query optimizer fails due to heavy or complex workload; it assists by stabilizing the system and generating the query execution plans. The discussed approach in Chaudhuri et al. (2007)

is limited to handle single query execution plan (QEP) and does not consider parallel QEPs. There is no dynamic way in proposed technique to maintain the restart plan during the modification of source records (past executed records). The technique proposed in Chandramouli et al. (2007) solves the optimization problem by using mixed-integer programming; however after query resumption, the technique does not re-optimize the given query. As compared to the previous approaches, this technique allows the suspension of the whole query, due to this memory wastage is less as compared to previous techniques. Schroeder et al. (2006) provided a mechanism that selects the right MPL value on the basis of only two parameters, that is, disc and CPU. The technique does not consider the impact of queries on each other. The low priority transactions are executed only when there are no high priority transactions. Moreover, the paper does not provide a mechanism to give priorities to different transactions. The technique improves the high priority transactions by the factor of 12.1 while the low priority average suffer is about 16%.

Self configuration

Database tuning advisor (DTA) (Agrawal et al., 2004) is an automated offline physical database design tool in SQL Server 2005. DTA provides physical design for a given workload and recommends horizontal partitioning, materialized views (MVs) and indexes. It produces script for the implementation of recommended physical design. Whenever DTA encounter any workload, it provides the recommendation by performing four steps. First, workload is parsed and compressed. After that each query is selected from a given workload and by using cost based model it provides suitable candidate configuration. Then in merging best step, one physical design structure is selected among the candidates configuration created in the previous step. In last step, enumeration is done by taking the union of the candidates as produced in last two steps and at the end it provides the final physical database design using Greedy (M, K) search scheme.

In Oracle, Index tuning wizard enhanced as SQL Access Advisor (SAA) (Oracle Corporation, 2007) as shown in Figure 3. On the basis of current workload, SAA recommends the indexes (including bitmap, function based and B-tree indexes), MVs and partitioning of tables, indexes, partitions, and materialized views. It has found that in a very short time and effort, the already tuned system can be tuned through SAA. It takes the contents from SQL cache and after analysis selects the appropriate indexes and MVs with possible benefits. It also performs a quick tune using a single SQL statement and how to make/ change materialized views.

DB2 Design Advisor (Zilio et al., 2004) is a tool that

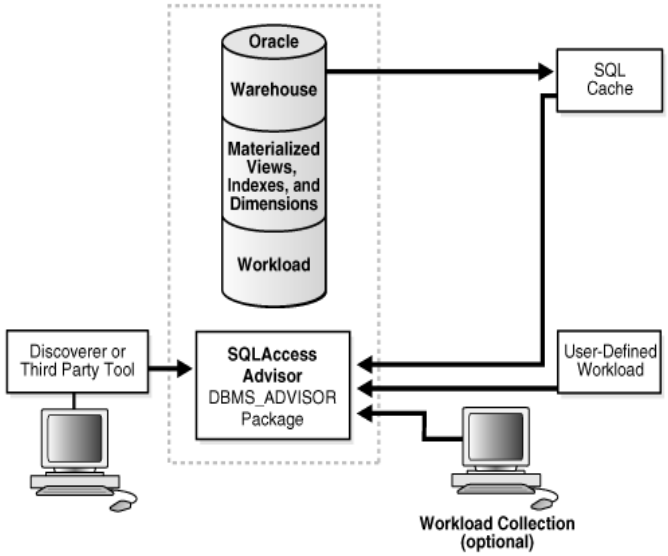

Figure 3. SQL Access Advisor [Oracle Corp., 2007].

automatically recommends physical design features for any provided workload that consists of set of SQL statements, which may include queries like updates, insert and delete etc. These physical design features may include selection of indexes, materialized views, shared-nothing database partitioning and multi-dimensional clustering of tables; only the DB2 Design Advisor recommends these four features. It provides a set of recommendations for selected features that reduces the total cost of workload within given disk space. Design Advisor provides candidate solutions and evaluates these solutions by making use of DB2 optimizer. Design Advisor has built in compression feature for scalability. Due to this feature, even the size of workload grows, the execution time does not increase exponentially unlike Index Advisor. The compression is performed only when heavy workload is encountered and Design Advisor feels that its analysis cannot be performed in finite time.

QUIET: Continuous Query-Driven Index Tuning (Sattler et al., 2003, 2007) is an online and predictive approach to handle workload. It selects the effective indexes that are already defined or appear to be beneficial. It predicts the benefit of each candidate index and selects the top beneficial index set. QUIET tracks the most recent queries that can be affected due to the candidate indexes. After selection of best indexes, if the benefit of new index set is greater than the pre-defined threshold then the older indexes are replaced with new ones. QUIET reduces the overhead of index creation by performing index creation and query execution at a time.

Continuous On-Line Tuning (COLT) (Schnaitter et al., 2007, 2006) is a framework that continuously examines the workload (incoming queries) and proposes the indexes to make physical design more valuable. COLT perform index selection in three stages, first builds a

model of current workload, second calculates the estimation of each candidate index and finally selects the best one. COLT collects statistics and allocates the profiling resources where these are required and reduces the overhead of online tuning. It performs extensively when workload change and slows down when workload is already well tuned. It regulates itself on the basis of heuristics. COLT is a separate component, which works parallel to the query optimizer. The proposed framework is implemented in PostgreSQL and found very effective to build indexes.

DTA (Agrawal et al., 2004) provides the integrated physical design recommendations for indexes, MVs and horizontal partitioning. DTA is scalable to large databases and can manage heavy workload using workload compression and reduced statistics creation techniques. It also allows DBA to specify his/her own manageability requirements with performance. For I/O, it provides maximum scriptability and customization through XML schema. However, DTA sometime generates bad results like when there are more than half of the columns in a table are multiple indexes. DTA provides the facility to DBA for iterative tuning and recommendations. So he can modify the previous recommendations and configurations for a certain workload until he is satisfied. DB2 Design Advisor (Zilio et al., 2004) is used to recommend the indexes, MVs, shared-nothing partitioning and multidimensional clustering of tables and over a benchmark baseline it has improves the performance of the workload. It has a built in module to reduce the current workload thereby enhancing the scalability. When the workload is compressed to maximum then there will be performance degradation; however design advisor reduces the maximum execution time with medium compression level. It has been proven experimentally that Design Advisor enhanced the performance up to 100%. SQL Access Advisor accepts the workload and provides the recommendations for indexes and Mvs that are beneficial for data access. The aforementioned tools are used for the physical database design only; such type of improvement and enhancements can also be made for logical design. Unlike DB2 Design Advisor, there is no such provision of clustering of tables in DTA and SQL Access Advisor. COLT (Schnaitter et al., 2007, 2006) is a predictive workload management approach where the benefit is calculated for individual indexes. The effectiveness of the index is calculated by subtracting the materialization cost from the predicting benefit. COLT has some overheads such as when selecting the best indexes for a given workload; it performs the what-if calls to query optimizer. However, COLT up to somehow reduces this cost by enforcing the limits on what-if calls. This limit can be increased or decreased according to the current workload. COLT claims to be online but it sets many parameters offline. Another major drawback in COLT is that it is limited to index and has no ability to suggests other physical design features such as materialized views, partitioning etc. Moreover, it suggests the single attribute indexes without considering the index correlation. While in real cases, multiple attribute indexes have much more benefits over the single attribute indexes.

Self-inspection

Query progress indicators are used to provide the step-by-step status of a query execution. Single Progress Query Indicator (Chaudhuri et al., 2005) is proposed with a graphical interface for relational DBMS that keep the track of work completion and estimate the remaining execution time. It starts its working by taking estimated cost from the query optimizer and calculates the remaining query execution time using statistics. It also monitors the query execution speed continuously and the remaining query execution time is estimated by dividing the estimated remaining query cost over the query execution speed. This indicator proposed a technique for single query and do not consider the impact of one query over the others one. So during the estimation of remaining time, this technique considers the load and query progress in isolation.

A multi-query progress indicator (Luo et al., 2006) has been proposed which represents the progress of running queries and considers the impact of queries on each other as oppose to the previous techniques of single query progress indicators. The technique of Multi-query PI works by considering the remaining execution time of concurrent queries with available statistics and predicts the future execution speed of incoming queries. On the basis of estimation, it can also predict the future queries; it has the ability to manage the current workload efficiently. The indicator not only provides the visualization of the running queries but helpful to manage workload efficiently. This technique takes workload management problems as input and provides their solution through the information as provided by Multi-query PI. The proposed technique for multi-query progress indicator is implemented in POSTGRE SQL and examined with remaining query execution time and workload management.

DB2 Query Patroller (QP) (IBM Corporation, 2003, Lightstone et al., 2002) is a management tool, which is used to streamline the requests according to available resources and workload. It is responsible to accept workload from user and analyze it. On the basis of analysis, it prioritizes and schedules the workload according to different query classes. A class is build by considering cost range and Multi Programming Level (MPL) threshold. Cost range is provided by the query optimizer that calculates the resource demands. MPL threshold is the maximum number of requests in a class that can execute in one time. Remaining queries are placed in a queue when the threshold level is reached

and are placed for execution in a class when threshold level falls. It also gives information to user about the status of the tasks. QP provides sufficient resources for given workload and by using profile (that is created by administrator) saturations for long terms queries can be avoided. QP controls the flow of requests proactively. It provides the information about the completion of request and finds trends of queries, workload of users as well as the frequently used indexes and tables. QP enhances performance by monitoring the system utilization, canceling or rescheduling the queries and identifying the trends of database usage. Query submitter uses QP to monitor submitted queries, store query result for future perspective and query submission customization. Submitter assigns higher priorities to some user so that in the class, their queries run with less delay. QP suspends high load queries so that they can be cancelled or scheduled to run after peak hours and track the query process. By performing these steps, the smaller queries may not stick and system recourses are used properly. QP is based on client and server architecture and consisting of three components. These components are Query Patroller server, Query Patroller Center and Query Patroller command line support. DBA uses QP to assign privileges of resources at user and system level.

REDWAR (RElational Database Workload AnalyzeR) (Yu et al., 1992) is an offline tool for DB2 environment, which is used for the characterization of SQL trace, provides structural information and statistics of the query under execution. REDWAR analyzes and classifies the data. During analysis, it uses statistical summaries (correlation, distribution, variation etc) and runtime behavior of the workload. The report generated by REDWAR can be used to plan the physical design and build benchmark workload. However REDWAR has no functionality to recommend physical design. It increases the efficiency of database system by identifying the criteria for a query.

A single-query progress indicator (PI) (Luo et al., 2004) often provides bad and wrong estimates. As in most of the concurrent queries execution, one query can significantly slow down the progress of other query. In single query PIs, greedy algorithm is used to speed up the process, where an optimal victim query is selected and the next optimal victim query is chosen. The process continues up to get all the victim queries. The technique in Luo et al. (2006) is the first proposal of a multi-query progress indicator. As compare to single query PIs, the Multi-query Progress Indicator considers the impact of concurrent queries over each other and predicts the incoming queries with priority and cost. The information provided by these indicators is helpful for workload management tools to take more intelligent decisions. The indicator is able to predict the accurate future queries even when initial estimates are wrong by detecting and correcting their estimates. It monitors the system at all times and manages the workload more dynamically. Multi-query PI is adoptive as it revises its decisions when it found some significant change as compared to predicted results. This adaptive behavior of Multi PI shows the consistency with the trends of automatic and autonomic computing. Query Patroller (IBM Corporation, 2003; Lightstone et al., 2002) monitors the given workload; perform analysis and prioritize it schedules for the incoming requests from the users. It limits the flow of long running queries to avoid saturation and ensures better resource utilization on the basis of profile (created by the administrator). REDWAR (Yu et al., 1992) is characterization tool for DB2 environment and assists the physical database design process. However, it does not provide the recommendations for physical design. Table 3 represents the techniques and model for workload self-inspection with other attributes.

Self-organization

Self-organization is the characteristic of DBMS to reorganize and restructure the layout of data and indexes dynamically. Research in the area of self-organization of workload management has been carried out by a number of researchers. Here, we are discussing some of these.

A tool Disk Array Designer (DAD) (Anderson et al., 2005) is used to configure the storage system. It is used to assist administrator in taking the decisions (using device model) about the physical design and design automation process. The algorithm used by DAD selects the best design among possible design choices using best-fit bin packing heuristic with randomization technique. This designer not only defines the array but also performs its configuration and storage for application data. DAD is evaluated by performing experiments over mix workload and found that it has the ability to handle critical configuration (low or high level) tasks. Moreover, it also produces near-optimal plan for the design of the storage system with speed and precision.

An adaptive QoS management technique is discussed by Krompass et al. (2008) where they used economic model that is used to handle individual request of BI and OLTP workload proactively. They provide a systematic way to arrange different requests by dividing these into different classes based on cost and time limit. They also proposed a model which calculates the cost of a request by differentiating underachieving and marginal gains of a Service Level Objective (SLO) threshold. The effectiveness of framework is determined by performing experiments on different workloads.

A Priority Adaptation Query Resource Scheduling (PAQRS) Algorithm (Pang an adaptive QoS management technique, 1995) is based on Priority Memory Management (PMM) Algorithm and deals with multi-class query workload. This algorithm reduces the missed deadlines according to the miss distribution defined by the administrator. The algorithm works by allocating

memory and assign priorities by considering resource usage, workload characteristics and performance experienced. Whenever the workload change, new multiprogramming level is calculated, memory reallocation and priority adjustment is done accordingly. Two techniques a miss ratio projection and resource utilization heuristics are used to calculate new MPL. In case of miss ratio projection method, previous MPL and miss ratio are used as parameters.

DAD (Anderson et al., 2005) searches the best design by using the best-fit bin packing heuristic with randomization and backtracking techniques; and estimates the performance of storage system through device models. Administrators can get the optimal solution by comparing their own storage system design to DAD design. Administrator may make changes in some storage part by DAD solution through answering the "what-if" questions. DAD provides the near optimal solution, as the storage design process is NP-hard problem. DAD provides the automatic adoption as well as designs to administrator. The DAD algorithm is limited to the design of storage system and can handle only 1-tier architecture. The framework provided for QoS in workload management (Krompass et al., 2008) is beneficial for OLTP and BI workload. The framework is scalable as it can implement the new workload management concepts with already previously implemented policies. The framework uses economic model with two economic cost functions (Opportunity Cost, Marginal Gains), where penalty information is added with the queries. The penalty information is used to make the efficient order of pending query execution. The scheduling policy used for OLTP workload in this framework is enhanced by considering the combine effect of priorities and service level objectives rather than considering merely priority. PAQRS (Pang et al., 1995) is used to schedule the complex type of workload and reduces the number of missed deadline thereby making the efficient use of system resources. It has bias control mechanism, which regulate the distribution of missed deadlines among different query classes. The MPL and memory is allocated on the basis of regular and reserve group quota. The priority of regular queries is higher than reserve queries. By doing this, PAQRS make adjustments between the miss ratio and the target distribution. PAQRS cannot handle transactions and is limited to workload consisting of mix queries. Its' performance degrades with the increased workload fluctuations. So the adoption mechanism of PAQRS is not up to the mark and need to be improved.

Self-prediction

DB Resource Advisor (Narayanan et al., 2005) is used to predict the response time and throughput dynamically. The advisor predicts the workload using what-if model without using configuration description. This will help advisor to guess about the status of the resources. The detailed architecture of Resource Advisor is shown in Figure 6. The authors identified the components required for self-prediction which are low level instrumentation, end to end transaction tracing and parameterized models of hardware resources. It provides accurate trends of response time in transactional tracing. They performed experiments on OLTP workload and observed that the Resource Advisor accurately predict the changes in workload.

The research devised a QoS controller for E-commerce applications (Menasce et al., 1999, 2003) that has the ability to manage work load by adjusting different configuration parameters. The adjustments are done through the QoS Controller by considering three performance goals such as average response time, average throughput and probability of rejection. Menasce et al. (1999) introduces a technique for characterization and generates workload models for E-commerce environment. They introduced a CBMG (Customer Behavior Model Graph) or state transition graph. This model graph represents similar navigational pattern for group of customers who perform same activities. Then the workload model and its parameters are identified and presents clustering algorithm for workload characterization. At the end, this technique is evaluated with different experiments. Menasce et al. (2001) improves the QoS level in E-commerce application by dynamically monitoring and tuning. This technique identifies best configuration parameters by combining hill climbing technique with analytical queuing model. They perform experiments to evaluate their technique by making comparison of QoS levels. In paper (Menasce et al., 2003), authors have design controllers that use analytic performance models with combinatorial search techniques. This modeling technique is used to identify the best configuration for the given workload. Their model is used to predict QoS parameters of workload. They show effectiveness of their technique through simulation and experiments.

The Resource Advisor (Narayanan et al., 2005)) is presented with a modular architecture in which CPU, buffer and storage models are integrated to predict the response time and throughput by identifying the required key components. Authors have taken the advantage of end-to-end tracing technique in visualization and understanding performance of the system. Resources are properly allocated on the basis of continuous monitoring. As compare to the resource advisor, current DBMSs lack of CPU, buffer and disk models. By using these models, Resource Advisor provides an accurate prediction and best performance results. When the size of buffer pool is lower then the Resource Advisor has high overheads per transaction. Due to continuous monitoring, the CPU overhead is 6.2% for online and 1.2% for offline execution. This overhead can be reduced by using some other appropriate techniques. Finally, the tool is evaluated through a prototype implementation in SQL Server however

it has the ability to incorporate with some other DBMS. There will be maximum session drops in the characterization technique (Menasce et al., 1999) when there are huge sessions or maximum load. Moreover the technique has no mechanism to manage or recover these drop sessions. The technique for QoS of E-commerce (Menasce et al., 2001) workload can handle dynamic workload and short-term fluctuations. The technique uses heuristic optimization with predictive queuing model and provides better results. It uses reactive approach rather than proactive. The techniques uses hill climbing technique for searching but when it stuck, the sub-optimal solution will be achieved. The QoS Controller maximizes the throughput up to 88% on average. When control interval level is less than or equal to 11, QoS controller do not exhibit any performance; however when control interval exceeds 11, performance increases up to 95%.

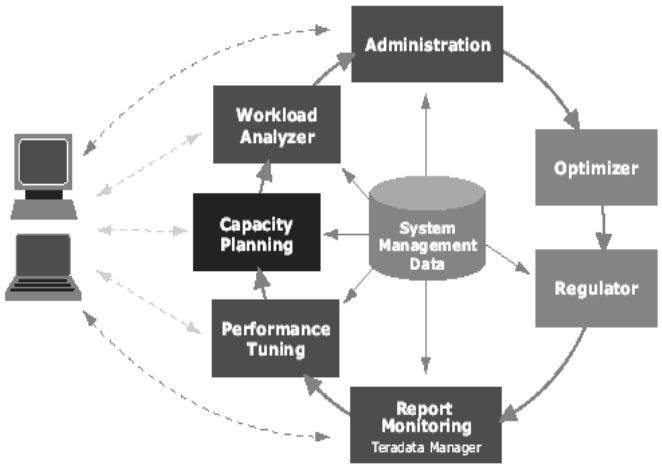

Figure 4. Teradata ASM Elements [Ballinger et al., 2002].

Self-adoption

Self-adoption or adaptation is the characteristic of ADBMS that is used to adopt the new changes according to the available resources and environment. In case of workload, self-adaptation is the way to adopt the given workload in an efficient and responsive way according to available resources and environment. Number of researchers studied workload adoption in DBMSs. The literature that fall in self- adoption category is discussed subsequently.

Resource Governor (Microsoft Corporation, 2007) is introduced in SQL Server 2008 that has the ability to manage workload and resources with business intelligence features. It provides a consistent, balanced and predictable response to users by imposing limits on resource consumption. It provides performance for concurrent workload by using a profile created by DBA. This profile contains the information about the resource limits and priorities for different workload groups. Resource governor allows setting timeout for certain queries and suggesting priorities without changing server settings. Resources are allocated by the resource governor as per requirements and priorities of different sub-department within the organization. So sub-department with high priority and requirements will get the large share of resources. The basic steps performed by the Resource Governor are to create resource pools, workload groups, classifier function, monitoring and adoption.

A framework is proposed (Bruno and Chaudhuri, 2007) for online tuning that examines current workload at all the time and then changes the physical design accordingly. When a workload is processed, their framework collects the information of the query execution plan (QEP), calculates its associated cost and then selects best QEP to alter the physical design. During the process, it uses query optimizer for acquiring the QEP. This framework also considers the index correlation during the process and by doing this it avoids the physical design oscillation.

In Teradata's ASM (Ballinger, 2002; Brown et al., 2008) workload is defined as a group that further consists of classification rule, MPL, exception and service level goals. Main elements of the ASM are shown in Figure 4. Classification rules are defined as attributes of the request that qualify the query to run under the workload definition. MPL is the number of queries that can be run in parallel under the workload definition. When limit exceeds, incoming requests are placed in the delay queue. Exception is used to control actions and produces when some abnormal behavior is found in the workload. The objectives of the workload are described in Service Level Goals (SLGs). ASM uses a preventive approach to workload change. The exceptions defined in the workload definition react with admission control of ASM. It cannot predict the resource demand required by each performance class, however maps the performance class with allocation group. The architecture of Teredata ASM consists of four phases, which are:

1. Queries are divided into classes by analyzing log and suggestions are made for workload definition.
2. Workload definitions are maintained by adjusting criteria, goals, and performance mappings.
3. On the basis of workload definition workload flow and priorities are regulated.
4. Examines the workload execution and improve the performance with respect to the goals. When sub-optimal performance is observed during the workload execution, four methods are used to improve performance. These methods are workload management, performance tuning, capacity planning and performance monitoring.

A framework for workload adaptation (NIU et al., 2006) has been proposed that has two components that are workload detection and workload control. The workload

detection finds the changes and provides information of the workload. The framework has also four functional components, namely workload characterization, performance modeling, system monitoring and workload control. The authors prove the effectiveness of their framework by implementing query scheduler and perform different experiments. Query scheduler can directly handle OLAP workload where as OLTP workload cannot be handled directly. They proposed a technique by using indirect control of OLTP through directly controlling OLAP workload. So due to this enhancement in query scheduler both workload can be handled to achieve performance goals. Authors improve the performance prediction process using Kalman filter as performance prediction plays vital role in workload adaptation. Kalman filter is very powerful filter and is used to make estimation of past, present and future states. This filter provides optimal solution for linear processes and sub-optimal solutions for non-linear processes. Through experiments, they obtained more accurate prediction results and observed less unpredicted SLO violations. In short, the research contribute by designing a general framework for performance oriented workload adaptation, prototype implementation of framework (Query Scheduler), a cost-based performance model for workload control plans and improves the accuracy of prediction through Kalman filter.

Resource governor (Microsoft Corporation, 2007) with BI features is used to manage workload in SQL Server 2008. It manages the given workload according to the profile created by the DBA. The profile contains the different information about the users such as their requirements, priorities etc. ASM (Ballinger, 2002; Brown et al., 2008) automate the workload up to significant level and focuses over ease of use with the help of monitoring and analysis. ASM is workload-centric as compares to other tools that mostly are system-centric. ASM changes the order of workload by using a preventive approach. ASM maps the performance class with allocation group and has no ability to predict the resource demand for each performance class. The experiment in Query Scheduler (NIU et al., 2006) is performed on stable workload, which is not suitable for dynamic environment where the workload changes rapidly such as in OLTP or OLAP. During the experiment, total cost of a query instead of detailed cost is used as a parameter that may generates wrong results. Moreover, it is confined to linear workload; however, in a real environment, most of the time workload is non-linear.

DISCUSSION

Previously, we have discussed the tools that are used to handle workload in different DBMSs and DWHs. Among these tools some are internal while the others are external that do not change the internals of DBMSs and DWHs. We have divided the entire work into workload type and autonomic perspectives. The workload type perspective reveals how much work is done in different types of workload while the autonomic perspective shows work in each autonomic characteristic.

Workload type perspective

Table 7 summarizes the research work done on workload management in database management systems and data warehouses by different researchers and vendors. The previous work on workload management is related with eight different workload types.

Autonomic perspective

Table 8 summarizes the research work done on the workload management in database management systems and data warehouses with respect to the autonomic characteristics by different researchers, practitioners and vendors.

CONCLUSION AND FUTURE WORK

The paper presents different aspects of autonomic computing such as its architecture, elements, characteristics and levels, autonomic DBMS, motivation towards autonomic DBMSs and autonomic workload management. To observe the current autonomic level in workload management, we have divided the available literature of workload management tools to self-* characteristics of AC. Tables 1 to 6 summarize the workload management with respect to autonomic characteristics on the basis of different parameters. This analysis shows the effectiveness of different available tools for workload management. Some advances on workload management in the context of autonomic computing have been done. However more efforts and improvements are essential on current as well as new workload management techniques and tools. In the future, we are planning to develop a framework for autonomic workload management that have the ability to handle all the tasks proactively.

REFERENCES

Agrawal S, Chaudhuri S, Kollar L, Marathe A, Narasayya, Syamala M (2004). Database Tuning Advisor for Microsoft SQL Server 2005. Proceedings of the 30th International Conference on Very Large Data Bases, pp. 1110–1121.

Anderson E, Spence S, Swaminathan R, Kallahalla M, Wang Q (2005). Quickly finding near-optimal storage designs. ACM Trans. Comp. Syst., 23(4): 337–374.

Ballinger C (2002). Introduction to Teradata's Priority Scheduler. http://www.teradatalibrary.com/pdf/eb3092.pdf.

Beg S, Naru U, Ashraf M, Mohsin S (2010). Feasibility of Intrusion Detection System with High Performance Computing: A Survey. Int. J. Adv. Comp. Sci., 1(1): 26-35.

Brown DP, Richards A, Zeehandelaar R, Galeazzi D (2008). Teradata Active System Management. http://www.teradata.com/t/page/145613/index.html.

Bruno N, Chaudhuri S (2007). An online approach to physical design tuning. Proceedings of the 23rd International Conference on Data Engineering, pp. 826–835.

Chaudhuri S, Kaushik R, Ramamurthy R (2005). When Can We Trust Progress Estimators for SQL Queries? Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 575–586.

Chaudhuri S, Kaushik R, Ramamurthy R, Pol A (2007). Stop-and-Restart Style Execution for Long Running Decision Support Queries. VLDB, pp. 735-745.

Chandramouli B, Bond CN, Babu S, Yang J (2007). Query Suspend and Resume. Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 557-568.

Dageville B, Das D, Dias K, Yagoub K, Zait M, Ziauddin M (2004). Automatic SQL tuning in Oracle 10 g. Proceedings of the 30th International VLDB Conference on very large databases, pp. 1098–1109.

Ejaz N, Baik SW (2011). Weighting low level frame difference features for key frame extraction using Fuzzy comprehensive evaluation and indirect feedback relevance mechanism. Int. J. Phys. Sci., 6(14): 3377–3388.

Horn (2001). Autonomic Computing: IBM's Perspective on the State of Information Technology. IBM J. Paper, pp. 1-38. http://researchweb.watson.ibm.com/autonomic.

Huebscher MC, McCann JA (2008). A survey of Autonomic Computing—Degrees, Models, and Applications. ACM 0360-0300/2008/08-ART7.

IBM Corporation (2003). DB2 Query Patroller Guide: Installation, Administration, and Usage.

Lightstone SS, Lohman G, Zilio D (2002). Toward Autonomic Computing with DB2 Universal Database. SIGMOD, 31(3): 55-61.

Luo G, Naughton JF, Ellmann CJ (2004). Toward a Progress Indicator for Database Queries. SIGMOD, pp. 791-802.

Luo G, Naughton JF, Yu PS (2006). Multi-query SQL Progress Indicators. Proceeding of the 10th International Conference on Extending Database Technology (EDBT), pp. 921–941.

Krompass S, Scholz A, Albutiu MC, Kuno H, Wiener J, Dayal U, Kemper A (2008). Quality of Service Enabled Management of Database Workload. IEEE Database Engineering Bulletin – DEBU, 31(1): 20-27.

Mateen A, Raza B, Hussain T, Awais MM (2008). Autonomic computing in SQL Server. 7th International Conference on Computer and Information Science, pp. 113-118.

Mehta A, Gupta C, Dayal U (2008). BI Batch Manager: A system for managing batch workloads on enterprise data warehouses. EDBT, 640-651.

Mehta A, Gupta C, Wang S, Dayal U (2009). rFEED: A Mixed Workload Scheduler for Enterprise Data Warehouses. ICDE, pp. 1455-1458.

Menasce DA, Almeida VAF, Fonseca R, Mendes MA (1999). A Methodology for Workload Characterization of E-commerce Sites. Proceedings of the First ACM Conference on Electronic Commerce, pp. 19–128.

Menasce DA, Barbara D, Dodge R (2001). Preserving QoS of E-commerce Sites through Self-Tuning: A Performance Model Approach. Proceedings of the 3rd ACM conference on Electronic Commerce, pp. 224–234.

Menasce DA, Bennani MN (2003). On the Use of Performance Models to Design Self-Managing Computer Systems. Proceedings of Computer Measurement Group Conf., pp. 1–9.

Microsoft Corporation (2007). Microsoft SQL Server 2005 Books Online. http://msdn2.microsoft.com/en-us/library/ms190419.aspx

Mumtaz A, Ashraf A, Shivnath B (2003). Modeling and Exploiting Query Interactions in Database Systems. CIKM2008, pp. 183–192.

Mumtaz A, Ashraf A, Shivnath B, Munagala K (2008). QShuffler: Getting the Query Mix Right. ICDE, pp. 1415-1417.

Mumtaz A, Ashraf A, Shivnath B (2009). Query interactions in database workloads. Proceedings of the Second International Workshop on Testing Database Systems (DBTest), USA.

Narayanan D, Thereska E, Ailamaki A (2005). Continuous resource monitoring for self-predicting DBMS. In International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, pp. 239–248.

Niu B, Martin P, Powwley W, Horman R, Bird P (2006). Workload adaptation in autonomic DBMSs. Proceedings of the 2006 Conference of the Center for Advanced Studies on Collaborative Research (CASCON'06). ACM Press, p. 13.

Oracle Corporation (2007). Oracle Database Performance Tuning Guide 11g. Release, 1(11.1).

Pang HH, Carey MJ, Livny M (1995). Multiclass Query Scheduling in Real-Time Database Systems. IEEE Trans. Knowl. Data Eng., 7(4): 533-551.

Parashar M, Hariri S (2005). Autonomic Computing: An Overview. Springer-Verlag Berlin Heidelberg, pp. 247–259.

Rhee A, Chatterjee S, Lahiri T (2001). The Oracle Database Resource Manager: Scheduling CPU Resources at the Application. High Perf. Trans. Syst. Workshop, p. 1.

Sattler K, Geist I, Schallehn E (2003). QUIET: Continuous Query-driven Index tuning. Proceedings of the 29th Very Large Database, pp. 1129-1132.

Sattler K, Luhring M, Schmidt K, Schallehn E (2007). Autonomous Management of Soft Indexes. Proceedings of the International Workshop on Self-Managing Database Systems, pp. 450-458.

Schnaitter K, Abiteboul S, Milo T, Polyzotis N (2007). On-line index selection for shifting workloads. Proceedings of the ICDE Workshops (SMDB), pp. 459–468.

Schnaitter K, Abiteboul S, Milo T, Polyzotis N (2006). COLT: Continuous On-Line Database Tuning. Proceedings of ACM SIGMOD, pp. 793–795.

Yu PS, Chen M, Heiss H, Lee S (1992). On workload characterization of relational database environments. Softw. Eng., 18(4): 347–355.

Zilio DC, Zuzarte C, Lightstone S, Ma W, Lohman GM, Cochrane R, Pirahesh H, Colby LS, Gryz J, Alton E, Liang D, Valentin G (2004). Recommending materialized views and indexes with IBM DB2 Design Advisor. Proceedings of International Conference on Autonomic Computing, pp. 180–188.

Zilio DC, Rao J, Lightstone S, Fadden S (2004). DB2 Design Advisor: Integrated Automatic Physical Database Design. Proceedings of 30th International Conference on Very Large Data Bases, pp. 1087–1097.