# Using Economic Models to Allocate Resources in Database Management Systems

Mingyi Zhang, Patrick Martin, Wendy Powley
School of Computing, Queen's University, {myzhang | martin | wendy}@cs.queensu.ca

Paul Bird
IBM Toronto Lab, pbird@ca.ibm.com

## Abstract

Resource allocation in database management systems is a performance management process in which an autonomic DBMS makes resource allocation decisions based on properties like workload business importance. We propose the use of economic models to guide the resource allocation decisions. An economic model is described in terms of business concepts and has been successfully applied in computer system resource allocation problems. In this paper, we present an approach that uses economic models to allocate multiple resources, such as main memory buffer space and CPU shares, to workloads running concurrently on a DBMS. The economic model enables workloads to meet their service level objectives by allocating resources through partitioning the individual DBMS resources and making system-level resource allocation plans for the workloads. The resource allocation plans can be dynamically changed to respond to changes in workload performance requirements. Experiments are conducted on IBM® DB2® databases to verify the effectiveness of our approach.

## 1 Introduction

The emerging trend of enterprises consolidating workloads onto a single database server makes the management of the diverse variety of workloads running on a database management system (DBMS) increasingly complex and costly. Workloads submitted by different applications, or from different business units, might have unique performance requirements which are normally described in terms of Service Level Objectives (SLOs) that must be satisfied. Workloads concurrently running on a server inevitably compete for shared system resources, such as CPU, main memory and disk I/O. As a result, it is possible that the workloads do not meet their SLOs by failing to acquire sufficient system resources.

To meet the performance requirements of workloads, database administrators (DBAs) need workload management policies and techniques to help manage the workloads running on a DBMS. An importance policy, which is an important factor in management, applies business metrics to classify workloads into multiple importance classes based on the workloads' business importance. A workload might be considered important if it is generated by an organization's CEO or if it is revenue producing. Less important workloads might be those pertaining to functions such as human resources or business report generation.

It is a challenge for DBAs to tune DBMSs according to the workload business importance policies as system-level metrics must be defined to measure customer expectations. Also, the high-level importance policies must be translated to low-level tuning policies in order to impact these metrics.

Autonomic computing suggests that DBMSs are capable of becoming self-configuring, self-tuning, self-protecting, and self-healing [2]. A key feature of autonomic DBMSs is policy-driven

management, which includes making resource allocation decisions based on properties such as workload importance. This feature enables the high-level business policies to be automatically translated into low-level system tuning actions. In addition, the feedback loop of autonomic DBMSs verifies that workload SLOs are met and initiates system reconfiguration if the system fails to meet the SLOs. With autonomic DBMSs, system administrators might concern themselves only with the high-level business policies, and let the system take care of the rest.

Resource allocation, as a workload management technique, manages individual DBMS resources and assigns the resource shares to workloads based on their business importance. With the allocated resources, the workloads can be guaranteed to meet their unique performance requirements when the workloads are simultaneously running on one database server. Economic models have been successfully applied in computer system resource allocation problems [1]. The models incorporate an implicit sense of business trades and concepts. Business importance policies can be easily implemented by economic models and a DBMS can in turn be indirectly managed by these policies.

The objective of our research is to provide a method to allocate resources based on workload importance policies in a DBMS as a step towards a self-managing system. The main contribution of the work is the specification of a framework for resource allocation approaches that utilizes an economic model to tune multiple-resource allocations for workloads having unique SLOs. We extend our previous work on allocating a single resource [4][8] to the case of multiple resources. A second contribution is a simulator of the economic model to validate the framework. We simulate a DBMS environment with several workloads of differing importance on a single database server. The workloads compete for limited system buffer pool memory space and CPU shares. Our simulator suggests resource allocations for the workloads as produced by the economic model. We verify the performance of the server by running the workloads on an appropriately configured DB2 system.

The remainder of the paper is organized as follows. Section 2 discusses related work. Section 3 describes the framework of the economic model and Section 4 explains the use of the economic model to allocate multiple DBMS resources to workloads based on their business importance. The validation of the framework of the economic model for multiple-resource allocations is presented in Section 5. We conclude and suggest future work in Section 6.

# 2 Related Work

A number of approaches for resource management in DBMSs have been proposed. In this section, we discuss techniques that are specifically relevant to policy-based or goal-oriented workload management in DBMSs.

Davison et al. [3] propose a framework for resource allocation based on concepts from microeconomics. The central element of the framework is a resource broker that schedules queries and allocates resources among executing operators to achieve system-wide performance objectives. They present a prototype broker that manages memory and disk bandwidth for a multi-user query workload.

Boughton et al. [4] present a framework for resource allocation by utilizing an economic model. In this framework, a limited number of buffer pool memory resources are allocated among competing workloads with different SLOs and levels of importance. A utility function is introduced to determine the maximum bids that resource consumers are willing to provide for the shares of the resource in order to achieve their SLOs. The framework automatically translates the high-level business workload importance policies to the low-level buffer pool tuning actions.

The frameworks of Davison and Boughton are both based on business concepts in order to reduce the complexity of resource allocation problems and to provide stability. In both frameworks consumers are assigned wealth to reflect their performance objectives and resource brokers are guided by the principle of profit maximization, which in turn maximizes the system-wide performance and workload objectives. Davison et al. use an admission policy to control resource contention by scheduling new queries for execution and use an allocation policy to control a query's bid for resources when the query is scheduled [3]. Boughton et al. use a high-level business importance policy to determine a workload's wealth and then the workloads bid for the shared resources. Wealthy workloads, therefore, achieve better performance than poor workloads.

Brown et al. [5] propose an algorithm to achieve a set of per-class response time goals for a multi-class workload through automatically adjusting DBMS multiprogramming levels (MPL) and memory allocations. They assume that the system is configured such that it is possible to satisfy the goals for all classes in steady state. Since it is difficult to predict response times as a function of MPL and memory, they avoid exhaustively searching the entire solution space to find optimal $<mpl_c, mem_c>$ pairs for the workloads to meet response time goals by proposing heuristics and a feedback algorithm to search the $<mpl_c, mem_c>$ pairs. In our study, we use a queueing network model [9] to predict the performance of a workload with a certain amount of resources, and apply a greedy algorithm to search for optimal resource pairs.

Niu et al. [6] propose a framework for workload adaptation in a DBMS. Their approach uses an adaptable admission control mechanism to ensure that multiple workloads with different importance levels meet their performance goals. The importance and performance goals of a workload are expressed in a utility function.

Schroeder et al. [7] present a similar framework to meet a set of OLTP workload quality of service (QoS) targets. The framework has two main components, the scheduler and the MPL advisor. The core idea of the framework is to maintain an upper limit on the number of transactions executing simultaneously within the DBMS. In their work, they divide transactions into different classes based on the transaction business importance, and obtain QoS targets and overall performance of the DBMS through choosing MPL levels.

The approaches of both Niu and Schroeder perform workload management in DBMSs based on the business importance policies and schedule queries outside the DBMS. They do not directly deal with resource allocations, and therefore, they do not require low-level resource management plans.

# 3 Resource Allocation Framework

As described earlier, resource allocation in DBMSs is a process of optimizing resource access among competing workloads to achieve their SLOs. The introduction of economic concepts to solve resource allocation problems could potentially reduce the inherent complexity. Our resource allocation framework, which is shown in Figure 1, provides the ability to dynamically allocate multiple resources to workloads in a DBMS based on the workload business importance. It extends our previous work on allocating a single DBMS resource to manage workloads [4][8] by replicating the structure for a single resource to multiple resources and adding a method to select the amounts for the individual resources. The economic model consists of a set of resources, resource brokers, a trade mechanism, and resource consumers. It represents a price based economy with a market mechanism of auctioning and bidding on shared resources.

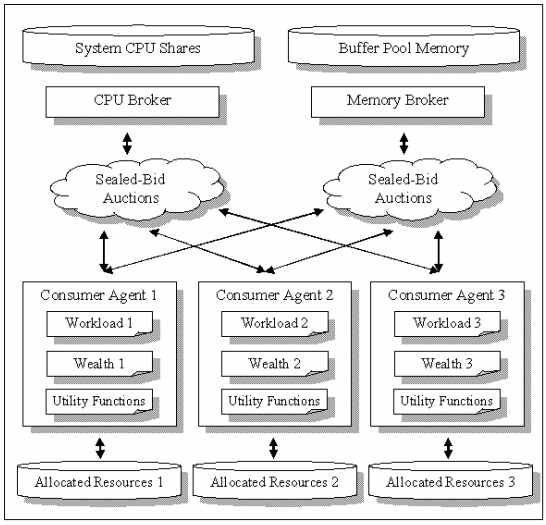


Figure 1: Economic Model Framework

## 3.1 Shared Resources

We investigate the allocations of two DBMS resources, namely buffer pool memory space and system CPU shares, through utilizing our proposed economic model. We choose these two resources since they are key factors in database performance management. We apply the economic model on a DBMS to simultaneously allocate these two resources to competing workloads based on the workload business importance polices to achieve the workloads' SLOs.

The buffer pool is an area of main memory in which data and index pages are cached to reduce disk access. The purpose of the buffer pool is to

improve database system performance by attempting to reduce the number of physical reads. For many workloads, a larger buffer pool size means that a database can achieve better performance. System CPU share is another main resource that contributes significantly to database system performance. A request in a database system needs a certain amount of CPU service to complete its work. If a database system obtains high CPU utilization, then the database can process more requests at a time and therefore achieves better performance.

This research extends our previous work of managing a single resource in a DBMS to the case of handling multiple resources. We have investigated the cases of buffer pool memory space and system CPU resources respectively [4][8]. In this study, we consider the case of allocating the two resources at the same time to competing workloads on a DBMS through utilizing the economic model.

## 3.2 Resource Brokers and the Trade Mechanism

Resource brokers allocate resources to consumer agents, who represent the workloads, through an auction-based economy. Each broker owns one type of resource and charges consumer agents for the use of the resources. Two resource brokers, namely the memory broker and the CPU broker, correspond to the buffer pool memory and CPU resources respectively. Consumer agents are assigned some amount of wealth to bid for the resource shares. The amount of wealth reflects a workload's business importance.

The principle behind the auction mechanism is that the resource brokers and consumer agents selfishly attempt to achieve their goals [1]. Resource brokers attempt to maximize their profits through conducting auctions to sell the resources they own and take the highest bids submitted by consumer agents. Consumer agents, on the other hand, attempt to maximize their workloads' performance by submitting the maximum bids they can afford to win auctions and therefore gather more resources. With resource brokers maximizing the profits, the optimal resource access among competing workloads and the SLOs are automatically achieved.

Several types of auctions, such as the Sealed Bid auction, English auction, and Dutch auction, are considered in an auction-based economy [1]. A sealed bid auction is used as the trade mechanism in our model since it is efficient and easily implemented. In sealed bid auctions, consumer agents are not aware of the amounts bid by other agents. Resource brokers collect the sealed bids submitted by consumer agents and select the highest bid as the winner, thus allocating the resources to the consumer agent. To maximize profits, resource brokers conduct auctions till there are no remaining resources or until there are no further bids. A lack of further bids indicates that the consumer agents have depleted their wealth or that their SLOs have been met using the current allocation. The price of a resource share in an auction is set by the highest bidder among the consumer agents. Resource prices, therefore, might vary through the process of all the auctions.

## 3.3 Consumer Agents

A consumer agent represents a workload running on the DBMS and competes for the shared resources with other consumer agents. Each agent executes queries in the workload and strives to meet the workload performance requirements. We consider several competing workloads as described earlier, thus there are the same number of consumer agents in the model correspond to the workloads respectively. A consumer agent typically consists of a workload, a certain amount of wealth, and a multiple-resource utility function. The interaction between the consumer agents and the resource brokers is through the sealed bid auctions, while there is no interaction among consumer agents or resource brokers in the framework.

A workload is conceptually divided into a series of *resource allocation intervals* of approximately equal number of requests [4]. At the beginning of each interval, all the resources are returned to resource brokers, and the consumer agent is re-assigned some amount of wealth. This amount is determined by two factors, namely the workload business importance level and the cost of the workload in the resource allocation interval. In our case, we use the total number of I/O operations of queries of a workload in an interval to represent the cost. If a consumer agent represents a workload $W$ with $K$ queries in a resource allocation interval, then the wealth assigned to the consumer agent, at the beginning of the resource allocation interval, could be calculated using equation (1)

$$Wealth(W) = imp_w * \sum_{i=1}^{k} cost(q_i) \quad (1)$$

where, $cost(q_i)$ is the estimated cost of *i-th* query of the workload $W$ in the resource allocation interval, and $imp_w$ is the workload business importance level called the *importance multiplier* which is defined in Section 3.4.

We use batch workloads in our studies, where all queries are known in advance, so we are able to estimate each workload's query I/O operations using the DB2 Explain utility [10].

Workload business importance levels can significantly affect the amount of wealth of consumer agents if the costs of the competing workloads are similar. A consumer agent might have much more wealth than others if the workload which it represents has the highest business importance, while a consumer agent might have much less wealth if its workload has less business importance. A wealthy consumer agent is more likely to win resource auctions, and therefore, its workload will be allocated sufficient resources to achieve high performance. The less important workloads associated with less wealthy consumers agents, on the other hand, might experience low performance. In a resource allocation interval, a consumer agent that wins an auction, gains the resource shares, but loses wealth. This ensures, therefore, that all consumer agents have a chance to win auctions [4].

In the economic model, a resource utility function helps consumer agents to determine the workloads' resource preferences. Bidding values of a consumer agent are determined by its wealth and marginal utility, that is, the difference in utility between two consecutive resource allocations [4]. If a consumer agent, representing the workload $W$, has the allocated resource pair $<c_{cpu}, m_{mem}>$, and it is bidding on $<x_{cpu}, y_{mem}>$ additional resources, then the marginal utility can be calculated with equation (2)

$$Mgl(w) = U((c+x)_{cpu}, (m+y)_{mem}) - U(c_{cpu}, m_{mem}) \quad (2)$$

where, $U(x, y)$ is the multiple-resource utility function which is defined in Section 4.3 Utility Function. Its maximum bid can be determined by the equation (3) [8]

$$Bid(w) = Mgl(w) * Wealth(w) \quad (3)$$

## 3.4 Workload Importance and the Importance Multiplier

In reality, workloads running concurrently on a database server are diverse and complex. They may have unique performance requirements and dynamic resource demands. For example, a transactional OLTP workload may have a high throughput requirement, while a decision support system workload may have a short response time goal.

We introduce the concept of an *importance multiplier* to capture the differences in the relative importance of competing workloads. As an example, consider a case where we wish to classify workloads into three business importance classes, which we label as *high importance*, *normal importance*, and *best effort*. We may assign a value of 1 to the importance multiplier of the best effort class and then express the degree of importance of the remaining classes relative to that value. For instance, if the high importance class has its importance multiplier as 10, then it is ten times more important than the best effort class. If the normal importance class has its importance multiplier as 5 then the class is five times more important than the best effort class, and in turn, the high importance class is two times more important than the normal importance class.

Through dynamically tuning importance multipliers and conceptually dividing workloads into a series of resource allocation intervals, we implement autonomic computing feedback loop feature in the economic model to verify the SLOs are met. At the beginning of each resource allocation interval, the consumer agents are reassigned wealth, and then the model re-allocates resources to the workloads through sealed bid auctions based on the consumer agents' wealth. With the workloads running on the DBMS, the model verifies that the workloads' SLOs are met. If necessary, the model could dynamically adjust the workload class importance multiplier values to indirectly change the submitting bids of the consumer agents in next resource allocation interval and in turn to affect the amount of allocated resources to the workloads in order to guarantee the workloads' SLOs are met.

# 4 Allocating Multiple Resources

The economic model, as described above, directly deals with multiple resource allocations in DBMSs based on workload business importance policies. In this paper we specifically consider simultaneously allocating buffer pool memory space and system CPU shares. There are three main components of our approach:

- **Resource model**: The resource model determines how to partition the buffer pool memory space and system CPU resources and what are reasonable total amounts of the two resources.
- **Resource allocation method**: The resource allocation method determines how to obtain optimal buffer memory and CPU resource allocations in order to benefit the workload and system-wide performance most and achieve the SLOs.
- **Performance model**: The performance model predicts the performance of a workload with certain amount of allocated resource shares in order to determine the benefit of the resources.

## 4.1 Resource Model

We assume each workload is assigned its own buffer pool so buffer memory pages can be assigned directly to a workload. CPU resources, on the other hand, cannot be directly assigned to a workload so we partition CPU resources by controlling the number of database agents that are available to serve requests on the database server, where we assume one database agent maintains one client connection. We conducted experiments to verify the relationship between the number of database agents and system CPU utilization for a workload and assume that the more database agents that are allocated to serve requests for a particular workload, the more CPU resources this workload receives [8].

We make the total amount of resources for their allocations as parameters in the resource allocation framework, so it could adapt to different system configurations of database severs. Once the parameter values of a specific database server are detected, then the framework could automatically suggest resource allocations to the workloads based on a given workload business importance policy on the database server.

## 4.2 Resource Allocation Method

Consumer agents attempt to acquire resources in order to maximize the performance of their workload. They must capture resources in appropriate amounts such that none of the resources become a performance-limiting, bottleneck resource. Consumer agents therefore need mechanisms to identify resource preferences of workloads in order to obtain optimal $<cpu_{opt}, mem_{opt}>$ pairs for the workloads.

In our approach, consumer agents use a greedy algorithm to identify the optimal $<cpu_{opt}, mem_{opt}>$ pairs for their workload. The resource allocation is determined iteratively. In an iteration of the algorithm, a consumer agent bids for a unit of the resource (either buffer pool memory or CPU) that it predicts will yield the greatest benefit to its performance.

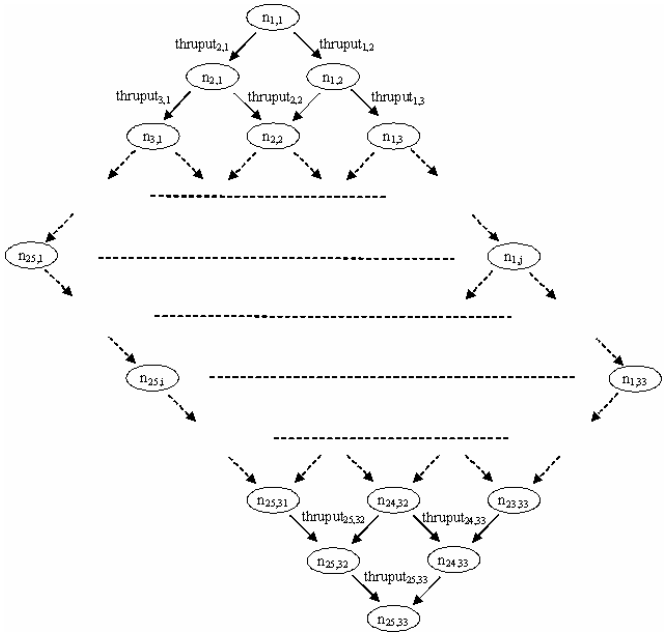


Figure 2: Buffer Pool Memory and CPU Pairs

Figure 2 shows a representation of the state space of the search algorithm for the case of allocating buffer memory and CPU resources to an OLTP workload presented in our experiments. The starting node, $n_{1,1,}$ represents the minimum allocation given to each consumer agent (one unit

of buffer memory and one unit of CPU). A consumer agent then traverses the directed weighted graph to search for the optimal $<cpu_{opt}, mem_{opt}>$ pair for its workload in order to meet the SLOs.

A node $n_{i,j,}$ in the graph, represents a possible resource allocation pair of $i$ units of CPU and $j$ units of buffer memory. The weight on the edge $(n_{i,j}, n_{i,j+1})$ represents the performance (in our case throughput) that could be achieved for the workload if the consumer agent acquires one more unit of buffer memory. Similarly, the weight on an edge $(n_{i,j}, n_{i+1,j})$ represents the performance that could be achieved if one more unit of CPU resource is acquired. A consumer agent always chooses to bid for the resource that will yield the best performance. If a consumer agent's bid is successful then conceptually it moves to the new node in the graph for the next iteration of the algorithm. If a consumer agent's bid is not successful then it remains at the same node in the graph. The graph has no cycles and the weight on each edge is determined by the performance model described in Section 4.3. The boundary nodes may have one or no outgoing edges when there is no more associated resource left. The proof that, for our problem, a greedy algorithm determines a global optimum is straightforward since the addition of more resources will always result in a performance improvement.

## 4.3 Performance Model

We model the DB2 DBMS used in our experiments with a simple queueing network model (QNM). This QNM is used to predict performance in each step of the greedy algorithm, that is, assign the weights to the edges of the graph in Figure 2. We use a closed QNM consisting of a CPU service center and an I/O service center. The CPU service center represents system CPU resources and the I/O service center represents buffer pool memory and disk system resources. We apply mean value analysis (MVA) [9] on the closed QNM to predict the performance of the DBMS with a certain number of database agents and a particular size of buffer pool.

For a specific database system and a specific type of workload, the average CPU service demand of the requests in the system is constant, and it can be determined by equation (4) [9]

$$S_{CPU} = (U * T)/C \tag{4}$$

where, $U$ is the CPU utilization, $T$ is the total workload execution time, and $C$ is the number of completed transactions in the experiment.

I/O service demand directly depends on the buffer pool size. If a buffer pool size is large, then it can reduce the number of I/O operations from disk system, which in turn reduces the average I/O service time of a request. On the other hand, if the buffer pool size is small, it can increase the number of I/O operations and so increase the average I/O service time of a request. The average I/O service demand can be expressed as a function of buffer pool memory size, using equation (5) [11]

$$S_{IO} = c * M^{b} \tag{5}$$

where, $c$ and $b$ are constant, and $M$ is buffer pool size. In the equation (5) the constants $c$ and $b$ can be determined by equations (6) and (7) respectively. Different database systems will have different $b$ and $c$ values.
For constant $b$:

$$b = \ln\left(\frac{S_{IO}1}{S_{IO}2}\right) \Big/ \ln\left(\frac{M1}{M2}\right) \tag{6}$$

For constant $c$:

$$c = (S_{IO}1)/(M1)^{b} \tag{7}$$

where, $S_{IO}1$and $S_{IO}2$ are I/O service demand values at buffer pool sizes of $M1$ and $M2$ buffer pages, respectively. $S_{IO}1$ and $S_{IO}2$ are determined through experimentation.

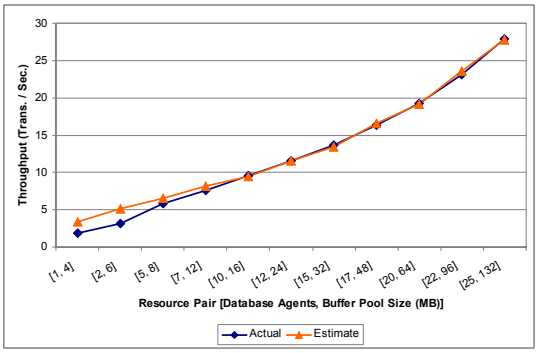


Figure 3: Validation of Performance Model

We use MVA on the closed QNM to predict throughput for an OLTP workload running on the database system with specific amounts of buffer memory and CPU resources. Figure 3 shows a validation of the model. It shows the predicted and actual throughput values for various resource pairs for the DBMS and TPC-C benchmark workload [12] described in section 5.

## Utility Function

In the resource allocation framework, a utility function [13] is a monotonically non-decreasing function, and it maps performance achieved by a workload with certain amount of resources into a real number $u$, $u \in [0, 1]$ [14]. If performance of a workload is high, then the utility of resources allocated to the workload is close to 1, while, if performance of the workload is low, then utility of resources allocated to the workload is close to 0. Consumer agents in the model employ a utility function to calculate marginal utilities, as defined in Section 3.3, to determine biding values in resource auctions.

We employ a utility function that normalizes the predicted throughput from the performance model relative to the maximum throughput that the workload could achieve when all resources are allocated to it. The utility function is given by

$$Utility(c_{CPU}, m_{MEM}) = MVA_{Throughput}(c_{CPU}, m_{MEM}) / X_{Max} \quad (8)$$

where, $MVA_{Throughput}(x, y)$ is the throughput provided by the QNM, $c_{CPU}$ is the number of database agents and $m_{MEM}$ is the number of buffer pool memory pages. $X_{Max}$ is the maximum throughput achieved by a workload with all resources allocated.

The marginal utility value reflects potential performance improvement, and by examining the marginal utility values, consumer agents can determine the preferred resource for the workloads. The maximum bid, as shown in Section 3.3, is the marginal utility multiplied by current wealth of a consumer agent, and says that a consumer agent is willing to spend the marginal-utility percentage of its wealth to bid for resources.

# 5 Validation and Experiments

Before integrating the economic model into a DBMS, we implement a simulator of the economic model outside of the DBMS to provide an experimental environment to validate the approaches proposed in the resource allocation framework. The validations address two main issues in the multiple-resource management studies, namely:

- Does the resource allocation framework generate the multiple-resource allocations for competing workloads on a DBMS so that the resource allocations match the workload business importance policies?
- Are the allocations resulting from the auctions sufficient for the competing workloads to meet their SLOs?

## 5.1 Experimental Environments

We apply the economic model using a single DB2 instance with three identical databases for three competing workloads from different importance classes. Each database is configured with one buffer pool and some number of database agents. Each database has one workload running on it, thus each workload has its own buffer pool and some system CPU shares while still having accesses to all the same database objects [4]. The economic model allocates buffer pool memory space and system CPU resources across the three identical databases based on a given workload business importance policy.

We experimentally determined the appropriate amount of total resources for the given configuration and set of workloads [8]. We selected a minimum amount of each resource where maximum system performance was achieved. We use 32768 buffer memory pages as the total buffer pool memory and 25 database agents as the total CPU resources.

The unit sizes of buffer memory and CPU are parameters in the economic model framework. We take 1000 buffer memory pages as one unit of buffer memory and 1 database agent as one unit of CPU resources in this study as these granularities give a reasonable workload performance increment and make the resource allocation efficient.

In this study, all experiments were conducted with DB2 Universal Database Version 8.2 [10] running on an IBM xSeries® 240 PC server with the Windows® XP operating system. The database server is equipped with two 1 GHz Pentium® 3 processors, 2 GB of RAM and an array of 11 disks. To eliminate the need to account for CPU parallelism in the QNM, as introduced in Section 4.3, we configure the system with one processor available.

The database and workloads are taken from the OLTP TPC-C benchmark [12]. The size of each database is 10GB. The three workloads are TPCC-like OLTP batch workloads stored in three script files. The workloads send transactions to the DBMS with zero think time during their run. Each workload consists of 120,000 requests based on the 5 different types of the transactions of TPC-C benchmark. A workload is divided into 12 resource allocation intervals. The start of each interval provides a checkpoint at which a resource allocation process takes place to reallocate resources to workloads based on their current importance levels and wealth.

The simulator is written in Java™ and the workload script files are used as the simulator input. The output of the simulator is resource allocations, that is, a list of the number of buffer memory pages and database agents for the workloads at each of their checkpoints. According to the output, database configuration commands are inserted into the workload script files at the checkpoints, allowing for a database to dynamically reconfigure during a run. By running the workloads on the DB2 database server, we verify that their SLOs are met. Before each run of the workloads, the databases are restored to their initial states. Buffer pool hit rate and workload throughput are taken to measure the workloads' performance. Each experiment was run five times, and the average of the five runs is reported.

## 5.2 Resource Allocations

A set of experiments was conducted to determine whether our approach generates the resource allocations which match a given workload business importance policy. The workloads are assigned one of three different importance classes, namely the high importance class, the normal importance class, and the best effort class, as described in Section 3.4. We experimented with three different sets of importance multipliers to validate the economic model. An importance multiplier set represents the importance multiplier values of the three importance classes in order of {best effort, normal importance, high importance}. The three sets used were {1, 1, 1}, {1, 5, 6}, and {1, 5, 10}. These configurations were designed to examine how resource allocation changes with the importance multipliers.

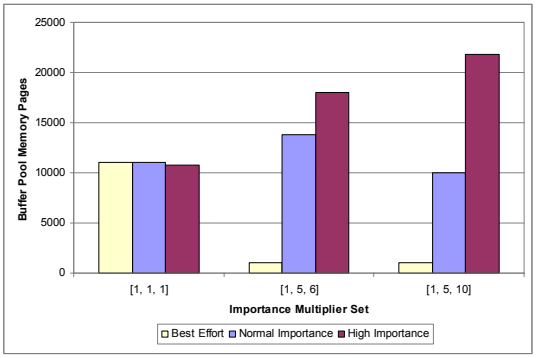


Figure 4: Buffer Pool Memory Allocations for Different Importance Multiplier Sets

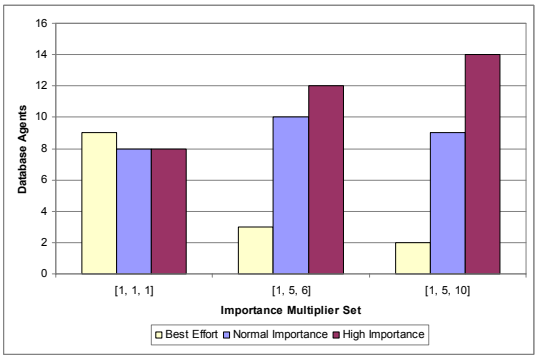


Figure 5: CPU Resource Allocations for Different Importance Multiplier Sets

Figure 4 and Figure 5 show buffer memory pages and database agents representing system CPU resources, allocations produced by the economic model using the three workload business importance multiplier sets. The workload importance multiplier set {1, 1, 1} represents the case where the three competing workloads are from three different business importance classes of equal importance. In this case, the three workloads are allocated approximately the same amount of buffer memory and CPU resources as shown in Figures 4 and 5. Using the importance

multiplier set {1, 5, 6}, the high importance and the normal importance classes are much more important than best effort class, and the high importance class is also slightly more important than the normal important class. When the economic model is used to allocate resources in this case, the high importance and normal importance workloads are allocated significantly more resources than the best effort workload, while the high importance workload is allocated slightly more resources than the normal importance workload. The set {1, 5, 10} represents the case where the high importance class is much more important than the normal importance class, and the normal importance class is much more important than the best effort class. In this case, the high importance workload is allocated more resources than the normal important workload, and the normal importance workload wins significantly more resources than the best effort workload.

We conclude that the resource allocations generated by the economic model for competing workloads match the workload business importance policies with more important workloads winning more resources than less important workloads.

## 5.3 Workload Performance

A set of experiments was conducted to determine whether the performance achieved by the workloads with allocated resources generated by the economic model match the business importance policy. The workload importance multiplier sets used in these experiments were the same as the sets used in the experiments shown in Section 5.2. We can therefore observe the achieved performance of the workloads with the allocated resources obtained in the first set of experiments. The workloads were run on DB2 using the resource allocations suggested by the simulator. Workload throughput and buffer pool hit rate were measured as performance metrics. The experimental results are shown in Figure 6 and Figure 7 respectively.

In Figure 6 we observe that the throughput of the workloads match the given business importance policies. When the workloads had the same business importance, namely applying importance multiplier set {1, 1, 1} on the economic model, the workloads achieve a similar throughput. When the workloads are of different business importance, namely applying multiplier sets {1, 5, 6} and {1, 5, 10} on the model respectively, then the important workloads achieve higher throughput than the less important workloads. The trend shows that the more important a workload is, the more resources it receives and hence, the workload performance is greater.

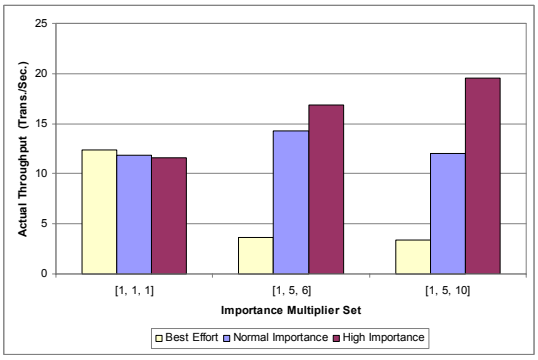


Figure 6: Workload Throughput for Different Importance Multiplier Sets

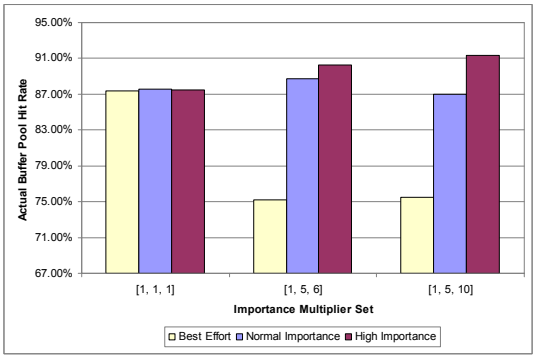


Figure 7: Buffer Pool Hit Rate for Different Importance Multiplier Sets

Figure 7 shows the buffer pool hit rate of the workloads for the three different importance multiplier sets.. The results show the same trend as the throughput results shown in Figure 6. When the workloads had the same business importance, then buffer pool hit rates are similar, while with different business importance levels, the important workloads achieve higher buffer pool hit rates than the less importance workloads. We notice that the difference in the hit rate between the high importance class and the normal importance class is small when the difference between their importance multipliers is large (that is, with the multiplier set {1, 5, 10}). This is likely due to fact that the total amount of buffer memory available for allocation is small, about 132MB. So even if the high importance workload obtained more buffer

memory resources, it could not achieve significantly larger buffer pool hit rate than the others [8].

Based on these results, we conclude that using the economic model for resource allocation is successful in obtaining workload performance that matches the given workload business importance policies.

# 6 Conclusions and Future Work

In this paper we present a framework that uses economic models to simultaneously allocate multiple resources, namely buffer pool memory space and system CPU shares, to workloads in a DBMS based on the workload business importance. In the resource allocation framework, we apply a queueing network model to predict performance of the workloads given a particular resource allocation. A greedy algorithm is used to find an optimal resource allocation for the workloads to maximize their performance to meet the SLOs. By presenting the economic model, we have shown a way that a high-level business importance policy can be automatically translated into a DBMS system-level multiple-resource tuning actions. Through a set of experiments we have validated the effectiveness of the framework.

An implementation issue of integrating the resource allocation framework into a DBMS is to predict the cost of a workload in an upcoming resource allocation interval. There are several possible approaches that could be used to address this issue. One approach is to estimate a workload cost for an upcoming interval based on the costs of the workload in previous intervals. Another approach is to allow consumer agents to request additional resources at any time by utilizing the feedback loop feature provided by the economic model to verify the workload SLOs are met, and not wait for the next resource allocation intervals. The third approach is to intercept the workload incoming requests by using the DBMS performance management component, such as DB2 Query Patroller [15], to estimate the query costs in a workload.

In the future, we plan to try exchange based economy to replace the current price based economy to allow the consumer agents to directly exchange their resources in order to simplify the trading mechanism in the economic model. We plan to add OLAP workloads to examine the effectiveness of the economic model with more complex workloads. In this study, we use batch workloads in which all queries are known in advance. As a next step, we plan to extend the economic model to accept random workloads to make the resource allocation framework more practical.

# Acknowledgements

This research is supported by IBM Canada Ltd., Toronto, Ontario, Canada, Natural Science and Engineering Council (NSERC) of Canada, and Ontario Centre of Excellence for Communication and Information Technology (OCE-CCIT).

# About the Author

**Mingyi Zhang** is an MSc student in the School of Computing at Queen's University. His research interests include: resource management in DBMSs, autonomic DBMSs.

**Patrick Martin** is a Professor in the School of Computing at Queen's University. He holds a BSc from University of Toronto, MSc from Queen's University and a PhD from University of Toronto. He is also a Faculty Fellow with IBM's Centre for Advanced Studies. His research interests include: database management systems, web services and autonomic computing systems.

**Wendy Powley** is a Research Associate and Adjunct Lecturer in the school of Computing at Queen's University. She holds a BA in psychology, a BEd, and an MSc in Computer Science from Queen's University. Her research interests include: database management systems, web services and autonomic computing systems.

**Paul Bird** is a Senior Technical Staff Member in the DB2 Database for Linux®, UNIX®, and Windows development organization within the Information Management group of IBM. His areas of interest include: workload management, security, monitoring, and general SQL processing.

# References

[1] D. F. Ferguson, C. Nikolaou, J. Sairamesh, Y. Yemini. "Economic Models for Allocating

Resources in Computer Systems". In Scott Clearwater, Editor, Market-Based Control: A Paradigm for Distributed Resource Allocation, Scott Clearwater. World Scientific, Hong Kong, 1996.

[2] J. O. Kephart, D. M. Chess. "The Vision of Autonomic Computing". IEEE Computer, Volume 36, Issue 1, pages 41-50, 2003.

[3] D. L. Davison, G. Graefe. "Dynamic Resource Brokering for Multi-User Query Execution". In Proc. of the ACM SIGMOD Intl. Conf. on Management of Data pages 281-292, 1995.

[4] H. Boughton, P. Martin, W. Powley, and R. Horman. "Workload Class Importance Policy in Autonomic Database Management Systems", Seventh IEEE Intl. Workshop on Policies for Distributed Systems and Networks, pages 13-22, London, Canada, June 5 - 7, 2006.

[5] K. P. Brown, M. Mehta, M. J. Carey, and M. Livny. "Towards Automated Performance Tuning for Complex Workloads", Proc. of the 20th Intl. Conf. of Very Large Data Bases, pages 72-84, Santiago, Chile, Sept. 1994.

[6] B. Niu, P. Martin, W. Powley, R. Horman, and P. Bird. "Workload Adaptation in Autonomic DBMSs", Proc. of the 2006 Conf. of the Centre for Advanced Studies on Collaborative Research, Article No. 13, Toronto, Canada, Oct. 2006.

[7] B. Schroeder, M. Harchol-Balter, A. Iyengar, and E. Nahum. "Achieving Class-Based QoS for Transactional Workloads", Proc. of the 22nd Intl. Conf. on Data Engineering (ICDE'06), page 153, Apr. 03 - 07, 2006.

[8] M. Zhang. "Using Economic Models to Tune Resource Allocations in Database Management Systems". M.Sc. Thesis, School of Computing, Queen's University, 2008.

[9] E. Lazowska, J. Zahorjan, G. S. Graham and K. C. Sevcik "Quantitative System Performance: Computer System Analysis Using Queueing Network Models". Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1984.

[10] IBM DB2 Universal Database Version 8.1 Administrative Guide: Performance, 2003.

[11] H. Zawawy, P. Martin, H. Hassanein. "Supporting Capacity Planning for DB2 UDB", Proc., of the 2002 Conf. of the Center for Advanced Studies on Collaborative Research, Toronto, Canada, 2002.

[12] Transaction Processing Performance Council. http://www.tpc.org/tpcc/

[13] W. E. Walsh, G. Tesauro, J. O. Kephart, R. Das. "Utility Functions in Autonomic Systems". In Proc. of Intl. Conf. on Autonomic Computing (ICAC'04), pages 70 - 77, New York, USA, May 17 - 18, 2004.

[14] G. Pacifici, M. Spreitzer, A.Tantawi, and A.Youssef. "Performance Management for Cluster Based Web Services", IEEE Journal on Selected Areas in Communications, Volume 23, Issue 12, page 2333-2343, Dec. 2005.

[15] IBM DB2 Query Patroller Guide: Installation, Administration, and Usage, 2003.

# Trademarks