

Automatic Performance Diagnosis for Changing Workloads in DBMSs

Darcy G. Benoit

*Jodrey School of Computer Science, Acadia University
Wolfville, Nova Scotia, Canada B4P 2R6
darcy.benoit@acadiau.ca*

ABSTRACT

Database performance is directly linked to Database Management System (DBMS) resource allocation. Complex relationships between DBMS resources make problem diagnosis and performance tuning difficult, time-consuming tasks. Database Administrators are currently required to retune the DBMS as databases grow and workloads change. Performance can be increased and cost of ownership reduced by automating the tuning process, starting specifically with the diagnosis of resource allocation problems. In this paper, we overview our automatic diagnosis framework designed to determine resource problems. We present our results demonstrating the ability to correctly identify system bottlenecks for a generic On-Line Transaction Processing workload when new transactions are added to the workload.

1. Introduction

A DBMS is a complex collection of subsystems that each performs a specific task and competes for system resources allocated by the DBMS. Difficulty in performance tuning begins with determining which resources need to be adjusted to solve the performance problem. In this paper we propose a method for automating the diagnosis process.

DBMS performance tuning is considered a non-trivial task [8][13]. DBAs skilled in performance tuning are scarce and expensive to employ [10]. As databases increase in size and complexity, manually performance tuning becomes “impractical” [7]. Several calls for the automation of the diagnosis and tuning processes have been made in recent years [5][7][8][10][11]. Automating the diagnosis process is the first step in allowing the DBMS to manage its own resources, reducing the amount of time that a DBA must spend attending to a system.

DBMS tuning involves two distinct tasks: diagnosis and resource adjustment. Diagnosis involves determining which resource(s) is responsible for the performance problem. Resource adjustment

(or “tuning”) involves altering resource allocations to achieve better performance. Automation would allow the DBMS to quickly achieve peak performance at a reduced cost by removing most human interaction.

Section 2 describes DBMS diagnosis literature. Section 3 overviews the diagnosis framework and models. Section 4 presents results of our system with a changing OLTP workload. Results are discussed along with the effectiveness of the diagnosis algorithm. Section 5 concludes by summarizing the results of the research and presenting additional areas of research in the area of automatic diagnosis.

2. Related Work

Related literature has provided information on previous work in the area of automating resource diagnosis. Weikum *et al* address the need for automatic memory management in data servers [15]. The paper surveys the possible approaches to memory management such as the self-tuning of cache memory and exploiting distributed memory and speculative prefetching for data and web servers. Work by Parekh *et al* assesses the application of “control theory to the evaluation of controllers” used for software management [12]. Hart *et al* propose a method to isolate performance problems of systems where performance data is stored in multidimensional databases (MDDBs) [9]. Bigus *et al* have proposed a generic agent for automated performance tuning [6]. The generic agent is designed to support tuning for systems where no prior knowledge is known to systems where effective resource controllers exist. Work by Martin [11] exists in the area of autonomic bufferpool management. Brown *et al* [7] has introduced the area of goal-oriented resource management for DBMSs, and has contributed in the area of memory management. Benoit *et al* [1][2][3][4] have introduced a generic feedback framework to be used for autonomic resource management in software systems.

3. Diagnosis Framework

We begin our diagnosis system by modeling both the workload and resources. Our resource model contained information from DBMS documentation about the relationships between various high-impact resources. Our resource model is a directed graph which shows that the adjustment of one resource has an effect on other resources in the system. The DBMS itself was modeled by running a workload against the system and adjusting each of the high-impact resources in pre-defined steps. As each resource was adjusted, performance information was monitored and stored. We monitored *indicator variables*, such as the buffer pool hit rate, and stored this information for later use. This information was used for workload modeling, where the importance of various resources and indicator variables to generic workloads were noted. For example, OLAP workloads usually require large sort spaces while OLTP workloads usually do not, making sort overflows an important indicator variable for OLAP workloads. Resource information was collected manually using documentation for the DBMS [13].

The workload and resource models are used in conjunction with a series of diagnostic rules to create the diagnosis tree used to diagnose the workload. Diagnosis is performed in a feedback loop where the performance of underlying resources are used to gauge the overall performance of the system. The feedback loop is shown in Figure 1.

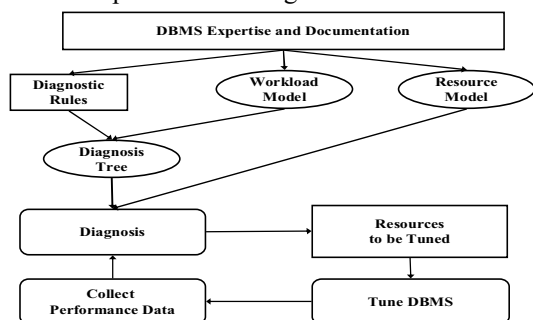


Figure 1 - Diagnosis overview

The core of the diagnosis process is the use of the diagnosis tree and the resource model. System diagnosis is accomplished by traversing the diagnosis tree. Starting at the root node of the tree, questions are posed about DBMS performance at each node. The performance of resources and indicator variables is compared to threshold values from the workload model. The diagnosis tree will be traversed to either the left or the right depending on the result of the comparison. This is continued until a leaf node is reached. It is important to note that the decision made

at each node is done with the knowledge of the decisions previously made during the tree traversal, so there may be more than one node in the decision tree that asks a question about a particular indicator variable or resource, although only one will be traversed. The leaf node contains a list of resources to be considered for tuning. At this point in time, our naïve tuning algorithms simply adjust the values of the resources listed in the leaf node. It is expected that further research will result in intelligent tuning algorithms that will make more complex resource adjustment decisions. Further details of this implementation can be found in [2].

4. Testing

To insure a realistic workload and database were used for our experiments, an un-audited TPC-C workload was used, measuring the number of “new order” transactions per minute, known as “tpmC” or “*Transactions Per Minute C*” [14]. Performance information was collected during each workload run. A naïve tuning strategy was used to determine how diagnosed resources were adjusted. Once a resource was identified by the diagnosis process, we adjusted that resource by a pre-determined amount.

The tests were run with two different workloads. The un-audited TPC-C benchmark was used for the initial workload. A new transaction was then added to the workload to simulate workload change. These methods were used in tandem to determine if the diagnosis system was able to automatically diagnose the problem resources.

4.1. Experiment and Results

We ran the OLTP workload with a buffer pool size of 100,000 4k pages, simulating a small database with sufficient memory for good performance. Table 1 presents data from the tuning process for the original workload. The “Changed Resource” column indicates the resource value that was altered from the previous run, “tpmC” shows throughput for that run, and “Diagnosis” provides the diagnosis for the run.

The results in Table 1 for Run 1 show that resource allocations were so poor that the throughput (tpmC) is zero due to poor lock resource allocations producing a deadlocked system. The diagnosis tree suggests tuning the locklist size parameter. The locklist parameter is adjusted from 40 pages to 60 pages, resulting in the diagnosis found in Run 2 of Table 1. Increasing the size of the locklist alleviates the deadlock problem and results in increased throughput.

Run 2 diagnosis suggests the number of I/O cleaners be adjusted. I/O cleaners are increased from

one to 10, resulting in decreased performance caused by an increase in the average lock wait time. Since the diagnosis system only considers the performance of the underlying system and not the throughput of the system, this is still considered an appropriate performance adjustment. Run 3 diagnosis suggests another I/O cleaner increase, resulting in a Run 4 performance increase that compensates for the decrease from Run 3. This process continues in Table 1 until the Run 10 diagnosis when the “tuned” node is returned. The algorithm has no further suggestions and the diagnosis and tuning process is complete.

We evaluate the success of the diagnosis algorithm by comparing the final throughput of the DBMS with tpmC values achieved by our expertly tuned system and a system tuned by the DB2 Tuning Wizard. The expertly tuned system represents the best possible throughput achieved when a DBA at Queen’s University at Kingston tuned the system.

The final measured throughput for the diagnosed configuration is 6270.86 tpmC. The throughput for the expert configuration is 6355.65 tpmC while the throughput for the wizard configuration is 4966.00 tpmC. The diagnosis system is able to tune the system to 98.67% of the expert throughput and 126.28% of the tuning wizard throughput. Throughput values are summarized in Figure 2.

#	Changed Resource	tpmC	Diagnosis
1	Initial Config	0.00	Locklist
2	Locklist=60	3897.18	num_iocleaners
3	Num_iocleaners=10	3150.18	num_iocleaners
4	Num_iocleaners=20	4254.41	dlchktime and/or locktimeout
5	dlchktime=50000	2984.18	dlchktime and/or locktimeout
6	dlchktime=10000	4593.00	num_iocleaners
7	Num_iocleaners=30	4258.88	dlchktime and/or locktimeout
8	Locktimeout=10	1418.29	dlchktime and/or locktimeout
9	dlchktime=5000	5986.94	num_iocleaners
10	Num_iocleaners=40	6270.76	Done

Table 1 - Diagnosis of the original.

A resource adjustment is considered valuable if the performance of the resource or indicator variable increases. For example, in Run 6 shown in Table 1, the throughput of the workload is 4593.00 tpmC. The diagnosis algorithm suggests increasing the number of I/O cleaners from 20 to 30 based on performance data indicating that the percentage of asynchronous writes was 85.0%, below the 95% threshold. Increasing the number of I/O cleaners to 30 results in 99.7% asynchronous writes for Run 7. Although

system performance decreased slightly, increasing the number of I/O cleaners is the appropriate action. The throughput decrease can be attributed to a shift in the performance bottleneck from physical data access concerns to deadlock concerns.

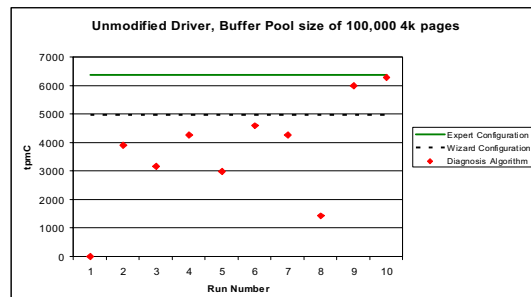


Figure 2- Throughput for unmodified workload.

Diagnosis time is an important factor for an automated diagnosis system. Our diagnosis algorithm runs for less than 30 seconds and resource adjustments takes less than 30 seconds. We collect performance data over a 20 minute interval. Diagnosis time can be further condensed by reducing the number of iterations needed. This can be done by using more intelligent tuning algorithms to avoid adjusting the same resource twice in a row, such as in Step 1 and Step 2 of Table 1. Such algorithms could reduce the number of iterations from 10 to 7 for the case in Table 1.

Another method to reduce the number of iterations needed to tune the DBMS is the use of resource trees. Resource trees expand the diagnosis space once a problem resource is identified. Expanding the diagnosis space to include related resources provides information that can be beneficial to the tuning process. By expanding the search space, it may be possible to identify key resources early and adjust them quickly.

Changed Workload

A changed workload was used to determine if the system was able to diagnose performance problems given a new transaction in the workload. We changed the workload by adding one new transaction intended to simulate a one-time query against the database. When the original database is used with the new, changed workload, the diagnosis system is able to tune the system to 100.1% of the expert throughput and 126.3% of the tuning wizard throughput. The result of the diagnostic and tuning process is shown in Table 2. Thirteen iterations were required for this diagnosis, a number that could be reduced to a few as 8 with intelligent tuning algorithms. Throughput values are summarized in Figure 3.

With the changed workload, the new transaction is not a part of the TPC-C benchmark, and as such, is

not reported in the throughput results. We have to separately consider the added transaction to determine if the diagnosis system was able to correctly identify the problem and propose an appropriate solution.

The new transaction added to the changed workload involves a large sort query on the same data that is being used by our initial workload. Sorting data uses allocated memory called a *sortheap*. When there is not enough memory in the sortheap, the DBMS must swap to disk, resulting in slower performance and a *sort overflow*. Reducing sort overflows is beneficial to the performance of the database management system.

In Table 2, we notice that the diagnosis results from Step 10 through Step 12 all deal with adjusting the sortheap and sort heap threshold (sheapthresh) to reduce the number of sort overflows. Performance increases are not noted in the tmpC measurement because the added transaction is outside of the benchmark. Separate testing of the sort query show that sort overflows cause the sort query to take more than 2.5 times as long to finish as the query without sort overflows. Sort overflows that occurred in Steps 10-12 do not occur in Step 13. Our diagnosis and performance tuning resulted in a reduction in sort query response times and increased the performance of the database.

#	Changed Resource	tmpC	Diagnosis
1	Starting Config	1.35	locklist Size
2	locklist = 60	2854.18	num_iocleaners
3	num_iocleaners=10	5198.88	num_iocleaners
4	num_iocleaners=20	3415.06	dlchktime and/or locktimeout
5	dlchktime=50000	4019.94	dlchktime and/or locktimeout
6	dlchktime=10000	4545.53	num_iocleaners
7	num_iocleaners=30	3770.76	dlchktime and/or locktimeout
8	locktimeout=10	1833.29	dlchktime and/or locktimeout
9	dlchktime=5000	5889.00	num_iocleaners
10	num_iocleaners=40	6134.00	sortheap and sheapthresh
11	sortheap=512 sheapthresh=1024	6155.29	sortheap and sheapthresh
12	sortheap=1024 sheapthresh=2048	6187.06	sortheap and sheapthresh
13	sortheap=2048 sheapthresh=4096	6202.65	Done

Table 2 - Changed workload diagnosis.

4.2. Summary

The results over the test cases have shown to be very positive in diagnosing DBMS resource problems and helping to improve overall database performance. Using a naïve tuning strategy, we correctly diagnose and tune each test case to within a few percentage points of the expert throughput and able to outperform the wizard throughput. By matching the performance of the expert configuration, we deem this method successful for the diagnosis of performance problems in our test DBMS.

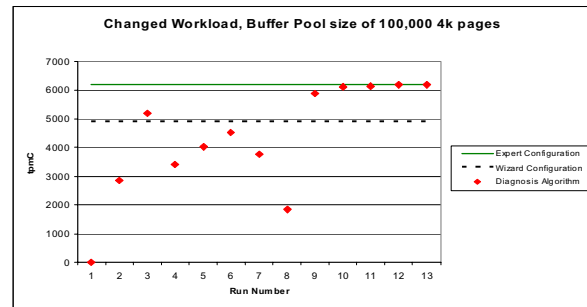


Figure 3 - Throughput for changed workload.

5. Conclusions

Achieving a high level of performance from a DBMS is a difficult task. The inherent competition that exists between DBMS tasks places a strain on various hardware and software resources. A balance must be achieved when allocating resources to obtain high levels of performance. The issue is further complicated because DBMSs do not stay in tune forever – performance may drop as data and workloads change. Automatic resource management removes human interaction to maintain high levels of performance. Automation involves both diagnosis and adjustment of resources to increase performance. With hundreds of resources to adjust, diagnosis is the starting point for self-managing DBMSs.

5.1. Contributions

Designing automated DBMS diagnosis systems is an important research issue. As the complexity of DBMSs and workloads increase, the need for such a system increases. The ability to automatically diagnose performance problems will reduce the need for constant attention from expensive DBAs and lower DBMS operating costs. Operating cost reductions and decreased complexity will encourage a more widespread use of DBMSs.

This paper demonstrates that a diagnosis framework can be constructed to automatically diagnose a subset of DBMS performance problems. The diagnosis model and the resource model are designed to fit within the Quartermaster framework [1] and provide a fully automated diagnosis and tuning system. The research contributions include:

- Section 3 overviews methods used to create diagnosis trees. The presentation of these methods sets the stage for further research into automatic generation of the diagnosis tree.
- The development of the diagnosis model demonstrates that the diagnosis process can be successfully automated. The results presented in Section 4 confirm the ability of a sample diagnosis tree to correctly identify system bottlenecks for a generic OLTP workload. The diagnosis model provides a basis for the creation of other diagnosis trees for different database workloads and DBMSs.
- The models provide the basis for a generic tuning structure. The models presented in this paper can be applied to other software systems where resource allocation is an issue.

5.2. Future Work

The research in this paper has many possible extensions. Relevant research topics include:

- automatic generation/updating of diagnosis trees
- convergence of the diagnosis process
- learning and self-modifying diagnosis trees
- application to embedded applications
- creation of intelligent tuning algorithms to increase the number of resources diagnosed by the system

The automatic diagnosis system presented in this paper is designed to run parallel to any DBMS. Integration of the diagnosis system into a DBMS will result in significant changes to the implementation. We believe the underlying principles of our diagnosis system will remain regardless of implementation.

6. References

- [1] Darcy Benoit, Wendy Powley and Patrick Martin. "Quartermaster: A Framework for Automatic Tuning of Database Management Systems", *Department of Computing and Information Science*, Queen's University, June 1999.
- [2] Darcy G. Benoit, "Automatic Diagnosis of Performance Problems in Database Management Systems" PhD Thesis, Queen's University at Kingston, Ontario, June 2003.
- [3] Darcy G. Benoit, "Automatic Diagnosis of Performance Problems in Database Management Systems", *Proceedings of the 2nd International Conference on*

Autonomic Computing, Seattle, WA, June 2005, pp. 326-327.

- [4] Darcy G. Benoit, "Automatic Diagnosis of Performance Problems for Shifting Workloads in Database Management Systems", *Proceedings of the International Advanced Database Conference*, San Diego, CA, June 27-30, 2005, pp. 90-95.
- [5] Phil Bernstein et al. "The Asilomar Report on Database Research", *SIGMOD Record*, Vol. 27, No. 4, pp. 74-80, December 1998.
- [6] J.P. Bigus, J.L. Hellerstein, T.S. Jayram, and M.S. Squillante. "AutoTune: A Generic Agent for Automated Performance Tuning", *Practical Application of Intelligent Agents and Multi Agent Technology*, 2000.
- [7] Kurt P. Brown, Manish Mehta, Michael J. Carey and Miron Livny. "Towards Automated Performance Tuning For Complex Workloads", *Proceedings of the 20th International VLDB Conference*, pp. 72-84, Santiago, Chile, 1994.
- [8] Surajit Chaudhuri and Gerhard Weikum. "Rethinking Database System Architecture: Towards a Self-tuning RISC-style Database System", *Proceedings of the 26th International Conference on Very Large Databases*, pp. 1-10, Cairo, Egypt, 2000.
- [9] David G. Hart, Joseph L. Hellerstein and Po C. Yue. "Automated Drill Down: An Approach To Automated Problem Isolation For Performance Management", *Proceedings of the Computer Measurement Group*, pp. 376-384, 1999.
- [10] David Lomet. Letter from the Editor-in-Chief, *Bulletin of the Technical Committee on Data Engineering*, Vol. 22, No. 2, page 1, June 1999.
- [11] Patrick Martin, Min Zheng, Hoi-Ying Li, Keri Romanufa and Wendy Powley. "Dynamic Reconfiguration: Dynamically Tuning Multiple Buffer Pools", *Proceedings of the International Conference on Database and Expert System Applications*, pp. 92-101, September, 2000.
- [12] Sujay Parekh, Neha Gandhi, Joseph Hellerstein, Dawn Tilbury, T.S. Jayram, and Joe Bigus. "Using Control Theory to Achieve Service Level Objectives In Performance Management", *Real-Time Systems*, Vol. 23, No. 1-2, pp. 127-141, 2002.
- [13] Tetsuya Shirai et al. "DB2 UDB V7.1 Performance Tuning Guide", *IBM Redbooks*, 2000.
- [14] *TPC-C Benchmark Specification*, <http://www.tpc.org>
- [15] Gerhard Weikum, Arnd Christian König, Achim Kraiss and Markus Sinnwell. "Towards Self-Tuning Memory Management for Data Servers", *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, pp. 3-11, 1999.