Automation Everywhere: Autonomics and Data Management

Norman W. Paton

School of Computer Science, University of Manchester
Oxford Road, Manchester M13 9PL, UK
npaton@manchester.ac.uk

Abstract. Traditionally, database management systems (DBMSs) have been associated with high-cost, high-quality functionalities. That is, powerful capabilities are provided, but only in response to careful design, procurement, deployment and administration. This has been very successful in many contexts, but in an environment in which data is available in increasing quantities under the management of a growing collection of applications, and where effective use of available data often provides a competitive edge, there is a requirement for various of the benefits of a comprehensive data management infrastructure to be made available with rather fewer of the costs. If this requirement is to be met, automation will need to be deployed much more widely and systematically in data management platforms. This paper reviews recent results on autonomic data management, makes a case that current practice presents significant opportunities for further development, and argues that comprehensive support for automation should be central to future data management infrastructures.

1 Introduction

Database management systems provide an impressive list of capabilities; they can answer complex declarative questions over large data sets, exhibit well defined behaviours over mixed workloads of queries and updates, present a consistent interface in the context of many changes to how or where data is being stored, etc. However, the development, deployment and maintenance of database applications remains a lengthy and complicated process. As a result, there are ongoing activities, in particular within the database vendors, to improve support for, or even to automate, tasks that have traditionally been carried out by skilled database administrators (e.g. [1, 10, 36]). In addition, as query processors are increasingly used in less controlled environments, there has been a growing interest in adaptive query processing, whereby queries can be revised during their evaluation to compensate for inappropriate assumptions about the data (e.g. [3, 26]) or to react to changes in the environment (e.g. [28]).

Several of these activities can be related to a broader activity in *autonomic computing*, which seeks to reduce the total cost of ownership of complex computing systems. Autonomic systems are often characterised by whether or not

they support *self-configuration*, *self-optimization*, *self-healing* or *self-protection* [20]. However, although several techniques recur in autonomic computing (e.g. [16, 34]), and there are even preliminary proposals for toolkits that can be applied to multiple problems (e.g. [17]), it cannot yet be said that there are well established methodologies for the development of autonomic systems. Relating this work to the state-of-the-art in databases, several basic techniques have been adopted in both areas, such as the use of control theory where it is applicable [35], but many proposals for autonomic behaviours seem to be developed largely in isolation, and to address specific problems rather than to make automation a central design goal in the development of complex infrastructures.

This is somewhat in contrast with the software architectures that underpin high-profile internet applications, such as Google or Yahoo. In such contexts, highly scaleable architectures have been designed that are less often associated with challenging systems management issues. Such scaleability is often achieved through the provision of judiciously selected functionalities, but raises the question as to whether there are interesting middle grounds between current data management and information retrieval systems that provide some of the benefits of both without incurring the design and management costs of classical database applications. This paper reviews current work on autonomic data management in Section 2, where it will be shown that there are a wide range of proposals, but that these can rarely be felt to integrate seamlessly to provide intrinsically adaptive data management infrastructures. Section 3 highlights several recurring limitations of current activities in autonomic data management, and makes some suggestions as to how they might be addressed. More speculatively, Section 4 suggests that automation should be a central tenant in the design of data management infrastructures, and that where this is the case, new areas may open up for the application of database technologies.

2 Examples: Automation in Data Management

Autonomic computing is motivated by the observation that computing systems are increasingly capable, pervasive and distributed, and that the cost of managing systems cannot be allowed to grow in line with their number and complexity. The same motivation underlies the desire to increase the role of automation in data intensive infrastructures, both to reduce management costs and to make performance more dependable in uncertain environments.

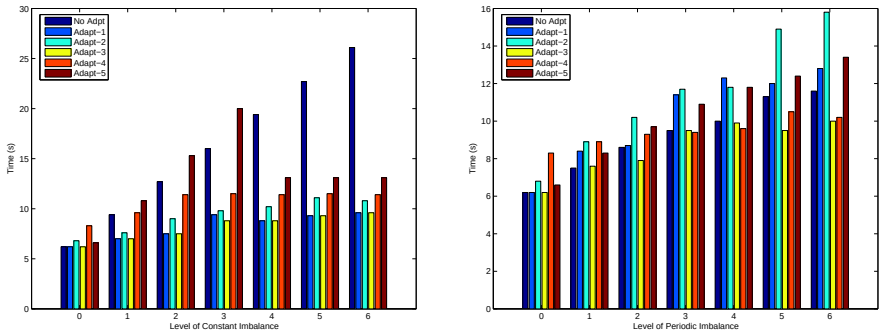
Much work in autonomic computing involves a control loop, in which feedback obtained by monitoring a system or the environment in which it is deployed leads to focused changes in the behaviour of the system. Such a model can be applied in general terms to a wide range of data management activities, and many aspects of data management are associated with some measure of autonomic behaviour. The following are examples of work to date:

Database Administration: The responsibilities of a database administrator include the classical self-management goals of autonomic computing men-

tioned in Section 1, namely configuration, optimization, healing and protection [11]. For self-configuration, a system may determine dynamically where to construct indexes, how to allocate memory to different functions, or which views to materialize (e.g. [36]). For self-optimization, a system may dynamically update the parameters used by cost functions, or automatically reorganise indexes to reduce fragmentation (e.g. [23, 25]). For self-protection, a system may dynamically limit the resources provided to a long-running query to reduce the impact of any one request on others. Overall, there is a substantial body of work on automating administrative tasks in database systems, and the major commercial products all provide tools to support and to automate various aspects of database administration.

Query Evaluation: Classically, query processing involves two distinct phases, namely optimization and evaluation. In optimization, alternative query plans are explored using a range of equivalence rules, and ranked on the basis of a cost model so that a preferred plan can be identified. In evaluation, the preferred plan is executed to yield the results of the query. However, this two-phase approach may yield significantly sub-optimal plans, for example if the cost model is based on partial or out-of-date statistics, or if the data is skewed in a way that is not taken into account by the optimizer. In adaptive query processing, decisions made by the query optimizer at compile time may be revised in the light of feedback obtained at query runtime [2, 13]. For example, specific proposals have been made that reoptimize queries, reusing at least some of the results produced to date, when selectivity estimates are shown to be inaccurate (e.g. [18, 33, 3]), or to rebalance load in parallel query evaluation (e.g. [14, 30, 31]). Overall, there is a substantial body of work on adaptive query processing, but at present few of the techniques have been incorporated into commercial database systems.

Data Integration: Most emphasis within the database community on data integration has sought to support the description of precise mappings between independently developed databases. Such mappings may be represented in many different ways (e.g. [8, 27, 32]), but are typically constructed and maintained manually. Both activities have proved challenging in practice, and at least partly as a result, database centred data integration products are not as ubiquitous as might have been anticipated. Various researchers have sought to develop schemes that automate the identification of mappings between models (e.g. [29]) or for change detection (e.g. [22]), in part accepting that the resulting mappings may be associated with different levels of confidence, in turn opening up the possibility that database integration technologies are used to provide lower cost and lower quality data integration, as in the vision for dataspaces [15]. However, there is wider interest in and benefit to be gained from inferring metadata in a distributed setting – for example, service descriptions are inferred from workflows in [5] – and higher-level data services such as discovery and integration often lean heavily on metadata, for which some measure of automatic creation and maintenance could significantly increase uptake. However, while there is now a growing body of work



(a) (b)

Fig. 1. Comparisons of the response times of several adaptive load balancing strategies for the same query in environments with different characteristics. (a) The query is running on three nodes, one of which has a high load for the duration of the run of the query, the level of which is varied in the experiment. (b) The query is running on three nodes, one of is subject to load spikes of duration 1s every 2s, the level of which is varied in the experiment. Most of the adaptive strategies improve significantly on the static strategy (*No Adapt*) in (a), but struggle to respond successfully to the much less stable environment in (b).

on automatic metadata capture, there has been less emphasis on incremental maintenance and refinement, as required by truly autonomic infrastructures.

3 Limitations

Although, as argued in the previous section, work on automation in data management is widespread, for the most part this work is quite fragmented. As a result, automation *per se* is not a major theme in the database community (the call for papers for VLDB in 2007 contains no mention of autonomic topics, although both SIGMOD and ICDE mention database tuning as an area of interest). A consequence is that there is little emphasis within the community on recurring pitfalls or potentially generic solutions. This section identifies some limitations in the state-of-the-art in the use of autonomic techniques in the database community.

Predictability: Autonomic behaviours involve intervention in the progress of an activity. As such interventions commonly incur some cost, may block ongoing activities while changes are made, and may discard partially completed tasks when changing the state of a system, there is certainly the potential for more harm to be done than good. For example, [3] describes circumstances in which an adaptive query processor may thrash by repeatedly identifying alternative strategies during the evaluation of a query, sometimes resorting to a previously discarded plan. An earlier proposal contains a threshold on

the number of adaptations it may carry out with a view to limiting the consequences of repeated adaptations in an uncertain setting [26]. Both [3] and [26] make particularly well motivated decisions as to when to adapt, and thus may be felt to be less prone to unproductive adaptations than many proposals. A comparison of several adaptive load balancing strategies [28] revealed circumstances in which all of the adaptive strategies did more harm than good, as illustrated in Figure 1, although all the techniques compared sometimes improved on static oprimization. Overall, however, the design, control and evaluation of adaptive techniques often seems to be as much an art as a science; we note, for example, that the developers of several of the proposals compared in [28] did not have a good understanding of the circumstances in which their techniques would perform better or worse than others or the non-adaptive case.

Methodology: Mature development activities tend to apply well defined methodologies that deliver predictable results. In autonomic systems development, a range of generic techniques have been explored, although their application in data management has been patchy; it is not always obvious which kinds of problem are most readily addressed by which techniques. Autonomic problems can often be characterised by a functional decomposition in which *monitoring*, *analysis*, *planning* and *execution* steps are identified, and proposals exist for toolkits that implement such components [17]. Such a framework, however, leaves open how the different components may be implemented, and in particular how decisions are made as to *what* changes are made and *when*. Applications of control theory to software systems are increasingly widespread [16], in which a model is developed of the behaviour of a system in response to changes in specific parameters. The resulting feedback control loops have been widely deployed for database tuning [35], but the changes made to an executing query in adaptive query processing are typically more radical than can be represented by changes to parameter values (e.g. many adaptive query processing strategies reoptimize queries at runtime, and thus the relationship between the state of the system before and after an adaptation is complex). Competitive algorithms [19], in contrast with control loops, focus principally on *when* to adapt, by trading off the risks of premature adaptation with the consequences of maintaining the status quo. However, although analyses have been developed that guarantee worst case performance of some adaptive algorithms relative to their static counterparts (this is where the word "competitive" comes from in "competitive algorithm"), it is not yet clear that they can be applied to typical adaptations proposed in data management. As such, significant work remains to be done to understand how best techniques developed in other domains can be applied to support adaptive data management systems.

Composability: Proposals for adaptive techniques in databases tend to address individual problems. For example, when automating database administration, techniques have been developed that determine which indexes to create or which views to materialise. However, such decisions tend to be made in isolation, even though the overall performance of a system depends

on complex interplays between these individual decisions. For the examples described, the need for views to be materialised may be affected by the presence of indexes, and vice versa. However, as such decisions tend to be made on the basis of mathematical models of system behaviour, and the design of the associated utility functions is problematic when multiple criteria must be taken into account [21], it remains an unsolved problem how to combine multiple autonomous components in a way that yields the desired overall behaviour [35]. Similarly, in adaptive query processing there may be ordering dependencies between adaptations. For example, in a parallel database that supports both partitioned and pipelined parallelism there may be adaptive techniques that remove imbalance in partitioned parallelism and bottlenecks in pipelined parallelism. However, a bottleneck may be able to be fixed by removing imbalance, and there may be no point removing an imbalance if there is a bottleneck elsewhere in the plan. As a result, the decision as to which adaptation should be applied in a specific context is not one that can always be made locally, even where specific problems with an evaluating query have been identified. Much of the state-of-the-art in adaptive databases involves the use of individual techniques in isolation, and few proposals have explored decision making for multiple strategies.

Semantics: Adaptive techniques change the behaviour of executing systems. As the executing systems are complex, and the changes made to their behaviour may be non-trivial, it may be desirable to have certain guarantees as to the behaviour of a technique. Such guarantees could, for example, place bounds on the worst case performance that adaptation could lead to [19], or demonstrate that the outcome of a request is sure to be unchanged by an adaptation [12]. However, rather few proposals for adaptive techniques are accompanied by formal characterisations of their semantics or behavioural guarantees, and generic techniques for specifying and reasoning about such systems seem not to be well established [4].

4 Opportunities

The limitations of adaptivity in database systems identified in Section 3 present certain challenges to the research community. In terms of *predictability*, a better understanding of benchmarking for adaptive systems and more comprehensive comparative studies may identify requirements that can inform the development of more effective development methods. In terms of *methodology*, the more systematic application of generic techniques, such as control loops or utility functions, may lead to a clearer understanding as to which kinds of problems can be supported by established and well founded techniques, and which stand to benefit from novel adaptive infrastructures. In terms of *composability*, foundational work is required to understand how to plan effectively in the context of multiple adaptive strategies, which in turn might hope to benefit from more systematic description of the *semantics* of adaptive behaviours. Such activities may in turn may be able to be applied to improve existing database systems or to support the development of new kinds of data management infrastructure:

Increasing the Manageability of Database Technologies: It is widely accepted that database technologies are labour intensive to administer, and that a significant portion of the cost of ownership of a database system is spent on administrators [23]. As discussed in Section 2, various aspects of database administration are now able to be automated, or at least supported by tools that monitor the use being made of a database installation. Some researchers have argued, however, that current commercial database systems are too complex, and that there is merit in developing data management platforms from collections of components, which themselves may be self-tuning [9]. Such components could include trimmed-down query processing capabilities, or storage managers specialised for specific kinds of data (e.g. video streaming). This proposal seems unlikely to be retrofitted to existing large-scale database platforms, but seems consistent with the recognised need for light-weight database systems, such as Apache Derby (http://db.apache.org/derby/), particularly for embedded use, or in support of distributed applications. To date, there has been little work on ensuring that lighter-weight database platforms are self-tuning; identifying the key features for which self-tuning is of benefit for such platforms, along with the provision of tools to support integration of self-tuning techniques across those features, seems like an important but viable activity.

Extending the Reach of Database Technologies: Although database technologies are dominant in many business sectors, web platforms that support data with different levels of structure largely ignore database models for description or languages for querying. Examples of structured data in the web include data behind web pages in the *deep web*, annotations in resources such as Flickr (http://www.flickr.com/), and online storage platforms such as Google Base (http://base.google.com/). The vision of dataspaces [15] seeks to bring database style querying to diverse data resources, whether or not they are managed using database management systems. Subsequent early proposals vary significantly in their context and emphasis, from integrating structured and unstructured data on a web scale [24], through the provision of enterprise level data access [6], to the management of an individual's data [7]. However, all such proposals share the need for automation in all the areas identified in Section 2, to enable low-cost data resource administration, efficient querying in unpredictable settings, and integration of data from potentially numerous sources. Typically, dataspaces are proposed for use in settings where certain sacrifices can be accommodated in the quality of query answers, as long as the cost of maintaining the data management infrastructure that provides those answers remains low.

5 Conclusions

Database management systems provide comprehensive facilities for creating, using and evolving potentially huge collections of structured data. As new requirements have been reflected in database systems over many years, the principal

database management systems have become increasingly complex, and thus expensive to manage effectively. As a result, the requirement for greater use of automation to support data management tasks has become increasingly evident. In the main, automation has been seen as something of an afterthought in most database systems, but the need to reduce the cost of deploying and maintaining data management infrastructures for data that is everywhere will necessitate a more central role for automation in future.

This paper has reviewed current practice in autonomic data management; the situation is that there has been widespread but largely uncoordinated exploration of the use of adaptive techniques for database administration, query evaluation, and data integration. Although individual proposals have been shown to be effective in specific contexts, the development of many adaptive techniques seems somewhat *ad hoc*; few proposals provide guarantees as to their worst case behaviour, and composition of different techniques into a comprehensively adaptive infrastructure remains largely unexplored. These characteristics reflect the fact that autonomic data management is rarely seen as a discipline in its own right, and to date much less attention has been given to the development of effective methodologies or to the understanding of good practice than to the development of solutions to specific problems. However, future data management platforms are likely to need to provide ever more robust behaviour in increasingly unpredictable settings, and thus automation is likely to be more central to their design than in current platforms. If database technologies are to be able to contribute effectively to the management and querying of ubiquitous data, automation will need to be ubiquitous too.

Acknowledgement: Research on autonomic data management at Manchester is supported by the Engineering and Physical Sciences Research Council, whose support we are pleased to acknowledge.

References

1. S. Agrawal, N. Bruno, S. Chaudhuri, and V. R. Narasayya. Autoadmin: Self-tuning database systems technology. *IEEE Data Eng. Bull.*, 29(3):7–15, 2006.
2. S. Babu and P. Bizarro. Adaptive query processing in the looking glass. In *CIDR*, pages 238–249, 2005.
3. S. Babu, P. Bizarro, and D. DeWitt. Proactive Re-Optimization. In *Proc. ACM SIGMOD*, pages 107–118, 2005.
4. H. Barringer and D. E. Rydeheard. Modelling evolvable systems: A temporal logic view. In *We Will Show Them! (1)*, pages 195–228. College Publications, 2005.
5. K. Belhajjame, S. M. Embury, N. W. Paton, R. Stevens, and C. A. Goble. Automatic annotation of web services based on workflow definitions. In *International Semantic Web Conference*, pages 116–129, 2006.
6. B. Bhattacharjee, J. S. Glider, R. A. Golding, G. M. Lohman, V. Markl, H. Pirahesh, J. Rao, R. Rees, and G. Swart. Impliance: A next generation information management appliance. In *CIDR*, pages 351–362, 2007.
7. L. Blunschi, J.-Peter Dittrich, O. R. Girard, S. Kirakos Karakashian, and M. A. Vaz Salles. A dataspace odyssey: The imemex personal dataspace management system. In *CIDR*, pages 114–119, 2007.

8. D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Information integration: Conceptual modeling and reasoning support. In *Proc. of the 6th Int. Conf. on Cooperative Information Systems (CoopIS'98)*, pages 280–291, 1998.
9. S. Chaudhuri and G. Weikum. Rethinking database system architecture: Towards a self-tuning risc-style database system. In *VLDB*, pages 1–10, 2000.
10. B. Dageville and K. Dias. Oracle's self-tuning architecture and solutions. *IEEE Data Eng. Bull.*, 29(3), 2006.
11. S. Elnaffar, W. Powley, D. Benoit, and P. Martin. Today's dbmss: Dow autonomic are they? In *Proc. 14th DEXA Workshop*, pages 651–655. IEEE Press, 2003.
12. K. Eurviriyanukul, A. A. A. Fernandes, and N. W. Paton. A foundation for the replacement of pipelined physical join operators in adaptive query processing. In *EDBT Workshops*, pages 589–600, 2006.
13. A. Gounaris, N. W. Paton, A. A. A. Fernandes, and R. Sakellariou. Adaptive query processing: A survey. In *Proc 19th BNCOD*, pages 11–25. Springer, 2002.
14. A. Gounaris, J. Smith, N. W. Paton, R. Sakellariou, and A. A. A. Fernandes. Adapting to Changing Resources in Grid Query Processing. In *Proc. 1st International Workshop on Data Management in Grids*, pages 30–44. Springer-Verlag, 2005.
15. A. Y. Halevy, M. J. Franklin, and D. Maier. Principles of dataspace systems. In *PODS*, pages 1–9, 2006.
16. J.L. Hellerstein, D.M. Tilbury, Y. Diao, and S. Parekh. *Feedback Control of Computing Systems*. Wiley, 2004.
17. B. Jacob, R. Lanyon-Hogg, D. K. Nadgir, and A. F. Yassin. *A Practical Guide to the IBM Autonomic Computing Toolkit*. IBM Redbooks, 2004.
18. N. Kabra and D. J. DeWitt. Efficient mid-query re-optimization of sub-optimal query execution plans. In *SIGMOD Conference*, pages 106–117, 1998.
19. A. R. Karlin. On the performance of competitive algorithms in practice. In *Online Algorithms*, pages 373–384. Springer, 1996.
20. J.O. Kephart and D.M. Chess. The Vision of Autonomic Computing. *IEEE Computer*, 36(1):41–50, 2003.
21. J.O. Kephart and R. Das. Achieving self-management via utility functions. *IEEE Internet Computing*, 11(1):40–48, 2007.
22. E. Leonardi and S. S. Bhowmick. Xandy: A scalable change detection technique for ordered xml documents using relational databases. *Data Knowl. Eng.*, 59(2):476–507, 2006.
23. S. Lightstone, G.M. Lohman, and D.C. Zilio. Toward autonomic computing with db2 universal database. *SIGMOD Record*, 31(3):55–61, 2002.
24. J. Madhavan, S. Cohen, X. Luna Dong, A. Y. Halevy, S. R. Jeffery, D. Ko, and C. Yu. Web-scale data integration: You can afford to pay as you go. In *CIDR*, pages 342–350, 2007.
25. V. Markl, G.M. Lohman, and V. Raman. Leo: An autonomic query optimizer for db2. *IBM Systems Journal*, 42(1), 2003.
26. V. Markl, V. Raman, D.E. Simmen, G.M. Lohman, and H. Pirahesh. Robust query processing through progressive optimization. In *Proc. ACM SIGMOD*, pages 659–670, 2004.
27. P. McBrien and A. Poulovassilis. Data integration by bi-directional schema transformation rules. In *Proc. ICDE*, pages 227–238, 2003.
28. N.W Paton, V. Raman, G. Swart, and I. Narang. Autonomic Query Parallelization using Non-dedicated Computers: An Evaluation of Adaptivity Options. In *Proc. 3rd Intl. Conference on Autonomic Computing*, pages 221–230. IEEE Press, 2006.

29. E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *VLDB J.*, 10(4):334–350, 2001.
30. V. Raman, W. Han, and I Narang. Parallel querying with non-dedicated computers. In *Proc. VLDB*, pages 61–72, 2005.
31. M.A. Shah, J.M. Hellerstein, S.Chandrasekaran, and M.J. Franklin. Flux: An adaptive partitioning operator for continuous query systems. In *Proc. ICDE*, pages 353–364. IEEE Press, 2003.
32. J. D. Ullman. Information integration using logical views. In *Proc. of the 6th International Conference on Database theory (ICDT'97)*, volume 1186, pages 19–40. Springer-Verlag, 1997.
33. T. Urhan, M. J. Franklin, and L. Amsaleg. Cost based query scrambling for initial delays. In *SIGMOD Conference*, pages 130–141, 1998.
34. W.E. Walsh, G. Tesauro, J.O. Kephart, and R. Das. Utility functions in autonomic systems. In *Proc. ICAC*, pages 70–77. IEEE Press, 2004.
35. G. Weikum, A. Mönkeberg, C. Hasse, and P. Zabback. Self-tuning database technology and information services: from wishful thinking to viable engineering. In *Proc. VLDB*, pages 20–31, 2002.
36. D.C. Zilio, J. Rao, S. Lightstone, G.M. Lohman, A. Storm, C. Garcia-Arellano, and S. Fadden. Db2 design advisor: Integrated automatic physical database design. In *Proc. VLDB*, pages 1087–1097, 2004.