Report: 3rd Int'l Workshop on Self-Managing Database Systems (SMDB 2008)

Introduction

Information management systems are growing rapidly in scale and complexity, while skilled database administrators are becoming rarer and more expensive. Increasingly, the total cost of ownership of information management systems is dominated by the cost of people, rather than hardware or software costs. This economic dynamic dictates that information systems of the future be more automated and simpler to use, with most administration tasks transparent to the user.

Autonomic, or self-managing, systems provide a promising approach to achieving the goal of systems that are increasingly automated and easier to use. But how can that be achieved? The aim of this workshop was to present and discuss ideas toward achieving self-managing information systems in an intimate, informal, and interactive environment.

SMDB 2008 was the second workshop organized by the Workgroup on Self-Managing Database Systems (`http://db.uwaterloo.ca/tcde-smdb/`) of the IEEE Computer Society's Technical Committee on Data Engineering. The Workgroup, which was founded in October 2005, is intended to foster research that enables information management systems to manage themselves seamlessly, thereby reducing the cost of deployment and administration.

Workshop Overview

The workshop was conducted in Cancun, Mexico on April 7, 2008, prior to the start of the International Conference on Data Engineering. The workshop's program committee consisted of the members of the SMDB Workgroup's Executive Committee plus four other well-known researchers who are leaders in the area. In response to the Call for Papers, the program committee received 19 submissions. Each paper was reviewed by 3 program committee members. Six papers were accepted to the Workshop, resulting in an acceptance rate of 32%. In an effort to make the Workshop as inclusive as possible, 4 more submissions were accepted as poster papers and given a shorter presentation time at the end of the workshop, for an overall acceptance rate of 53%. This year, we added an invited keynote speaker and a panel session featuring four distinguished researchers to summarize and comment upon the Workshop's presentations and discussions. The average attendance at the Workshop throughout the day was 40 participants.

Technical Program

The technical program was organized into 4 sessions: Welcome and Keynote Talk, Self-Healing and Self-Optimization, Physical Design and Virtualization, Poster Papers, and Panel and Wrap-up. Due to a travel delay, the keynote speaker presented after the second and third sessions. Links to the slides presented for each talk can be found in "Workshop Program" under SMDB 2008 at the Workgroup's web page (`http://db.uwaterloo.ca/tcde-smdb/`).

The first session contained three papers on ways to enable self-healing and self-optimization of databases. Nehme [1] advocated a comprehensive approach to self-managing systems, in which all aspects of systems management, performance, risk assessment, and availability are described and managed as part of a unifying self-healing framework. Most modern DBMSs have hundreds of configuration parameters, so it's impossible to evaluate all combinations of possible values. Debnath et al. [2] devised a practical approach to determining good configurations by exploiting a Plackett and Burman methodology that ranks queries based upon how sensitive they were to the extrema of the factorial design of all configuration parameters. Yellin et al. [3] extended traditional control theory concepts of "flux" to automatically balance the load of processors performing a join that is partitioned among them, taking into account the cost to response time of changing the partitioning on the fly when the load on some of the processors is perturbed. By adding heuristics to limit the frequency of

adaptation, they were able to reduce the number of specious changes caused by overly-adapting, as observed in the traditional "flux" technique.

The papers of the second session dealt with problems of physical database design and the increasing use of virtualization. Malik et al. [4] addressed the problem in the SkyQuery sky survey database of widely-varying ad hoc queries to tables having hundreds or even thousands of columns, few of which are referenced in any given query. Their solution was an adaptive, on-line vertical partitioning algorithm that improved upon an existing, off-line vertical partitioning algorithm (Autopart) and exploited some structure in the problem to prune the large solution space to a computationally tractable size. Minhas et al. [5] measured how much overhead the Xen hypervisor introduces when running a database (Postgres) workload. In a head-to-head comparison between a virtualized and "bare" operating system, the authors found significant overhead (tens of percent) introduced by virtualization when the buffer pool was warmed, but was much smaller (around 6%) when the buffer pool was cool. In fact, in some cases the I/O wait time was even lower with Xen, because pre-fetching in Xen's Dom0 is better than that in Postgres! Tata et al. [6] argued that physical design advisors might better be located on clients, exploiting server-based advisors if available but also dealing with the common case that either such server-based tools are unavailable or the prerequisite information for running them might not be available, e.g., before the data is loaded. They suggested ways to glean useful physical design information from what limited schema, data, statistics, and/or workload is available when design decisions must sometimes be made.

After lunch, John Wilkes of HP presented his keynote talk, "Utility functions, prices, and negotiation", which addressed the problem of designing in a principled way meaningful Service Level Objectives (SLOs), an important part of Service Level Agreements (SLAs). Wilkes described a technique that exploits the concept of utility functions, which measure some degree of "goodness" to those involved. In the 2-dimensional space of pricing vs. service outcome (e.g., throughput), utility indifference curves define contours of equal utility. The consumer chooses one such indifference curve, below which defines a "minimal acceptable utility" region. Similarly, the service provider chooses a (usually different) utility indifference curve, above which defines a region in which he is comfortable. Any intersection of the two regions defines an area within which negotiation on the SLO is possible. That negotiation, however, is much harder to characterize rationally or to quantify, because people sometimes react irrationally, are often averse to losses, and tend to overweight rare, extreme events.

The poster session contained the four poster papers with shorter presentations. Furtado et al. [7] described a prototype of a DBMS (based upon PostgreSQL) that uniformly reduced response times by up to 56% by continuously monitoring usage and adapting to meet quality of service (QoS) objectives. Voigt et al. [8] addressed the problem of off-line but dynamic physical database design, i.e., taking the order of arrival of queries into consideration. They modeled each query and a corresponding configuration (set of indexes) as a node in a state graph, which is huge but easily solved. To avoid the pitfall of over-fitting to a particular workload and the exact order of arrival, they simply limited the number of possible transitions. Sharaf et al. [9] described ASETS, a self-managing transaction scheduler that is formed as a hybrid between the optimal algorithm for low utilization and the optimal algorithm for high utilization. The combined algorithm uses an SLA to calculate a deadline for each transaction, then puts it on one of two ordered lists, depending upon how tardy it is or how much slack it has to make its deadline. Finally, Rizvi et al. [10] gave an overview of IBM's Balanced Warehouse (virtual) appliance for Business Intelligence workloads, which is composed of Balanced Configuration Units (BCUs), each a pre-configured, pre-tested unit that can deliver the performance required and allow incremental growth, but runs on non-proprietary hardware.

The last session had a panel format, with four distinguished panelists: Surajit Chaudhuri of Microsoft Research, Guy Lohman of IBM Almaden Research Center, Ken Salem of the University of Waterloo, and our keynote speaker, John Wilkes of HP. Each tried to respond to five questions in light of the day's presentations:

1. **Is completely self-managing achievable?** What are the biggest roadblocks to that, both technically and in gaining the trust of the user to enable the DBMS in "autopilot mode"? How do we avoid making life

worse for the administrator by adding more things that can go wrong? Is "less really more", i.e., is the only way to get simplified management by making things go away rather than have wizards set dials / thresholds?

2. **Can administration be standardized the way SQL querying has been standardized?** The success of relational DBMSs has been significantly helped by the standardization of SQL, but administration remains very different from one vendor to the next. Can / should administration be standardized somehow? Would this facilitate the emergence of client-side tools?

3. **How do we know when we've succeeded?** We're used to measuring performance, but how do we measure self-managing or ease of use? If we can measure it, how would we benchmark it? What aspects of self-managing can be realistically included in such a benchmark (i.e., is it possible to test automatic recovery from realistic failure modes)?

4. **How can self-managing tools function with incomplete information**, e.g. how can we initially configure a system without having a workload and/or database statistics? Are existing tools too sensitive to the workload, anyway? How do we reduce the overhead that these management tools and their information needs impose on the DBMS?

5. **Self-managing DBMS: who cares?** DBMS are not deployed in isolation. If self-managing DBMS are challenging, can we hope for a self-managing stack? If not, should we bother with self-managing DBMS? If so, how should DBMS fit in with end-to-end self-management?

Chaudhuri enumerated all the reasons why self-managing is so hard to solve (e.g., large search spaces of possible configurations, difficulty of diagnosing problems automatically, the limitations of query optimizers as modeling tools, ...), but also listed areas in which advances have been made, notably memory management, index selection, enabling "what if?" analysis, and establishing some fundamental principles. However, he warned that a unifying theory of self-managing was unlikely in the near term, and that progress would likely continue to be made incrementally on individual problems. He also noted that robustness of self-managing tools is extremely important to establish trust with users.

Lohman said that users certainly care about self-managing, but they don't trust features that aren't on by default, and the loss of trust due to a failure is hard to recover, quoting an actual incident with early automated Bay Area Rapid Transit trains. He was somewhat skeptical that complete self-managing was possible, due to the complexity (and hence brittleness) of our models, but clearly great progress is still being made. He cautioned that we too often rely on performance measures because they are familiar, rather than real measures of self-managing or ease of use, which have yet to be devised. Finally, standardization in the administration area remains elusive, because the data definition language (and its underlying storage model), unlike the query portions of SQL, were never standardized and hence have diverged. He concluded that we have succeeded to a degree, but are farther from our goal than we like to admit.

Salem emphasized that databases, while an important part of the problem, do not exist in a vacuum, but are part of a much larger ecosystem that includes hypervisors, operating systems, application servers, etc. What gets deployed are complete systems, not components, and these must be managed and tuned together as a system. The database is not the center of the universe.

Wilkes stressed the importance of trust, and the difficulty of earning it from humans, who aren't always rational. He noted that people are far better at dealing with exceptions and approximations than are machines, and systems can often ignore useful information. He felt that policies (rules) were not the answer, because there are too many of them that would need to be written. People will accept and trust automation when the benefits exceed the cost, and the worst case disasters are no worse than what would happen with a person in charge. Trust only comes from reassurance that the system will always "do the right thing", and only then will the human give

up control. Be sure never to take away that control without the human's permission, explain your automated decisions, and be wary of machine learning, which can be prone to inconsistencies, he advised.

Summary

Once again, the Workshop on Self-Managing Database Systems was extremely successful. Not only was attendance a bit higher than the previous year – despite the lure of Cancun's beach! – but so was participation through probing questions and lively discussion. The high quality of the papers and the enthusiastic interaction in the workshop demonstrate the vitality of research in self-managing information management systems.

The Workgroup on Self-Managing Database Systems would like to thank the participants and the organizers of the Workshop. They encourage anyone interested in making systems easier to manage to participate in the 2009 Workshop on Self-Managing Database Systems, which will be part of the International Conference on Data Engineering in Shanghai, China next spring.

References

[1] Rimma V. Nehme: Database, heal thyself, Proceedings of ICDE Workshops (SMDB 2008), pp. 4-10, Cancun, Mexico, 2008.

[2] Biplob K. Debnath, David J. Lilja, Mohamed F. Mokbel: SARD: A statistical approach for ranking database tuning parameters, Proceedings of ICDE Workshops (SMDB 2008), pp. 11-18, Cancun, Mexico, 2008.

[3] Daniel M. Yellin, Jorge Buenabad Chávez, Norman W. Paton: Probabilistic adaptive load balancing for parallel queries, Proceedings of ICDE Workshops (SMDB 2008), pp. 19-26, Cancun, Mexico, 2008.

[4] Tanu Malik, Xiaodan Wang, Randal C. Burns, Debabrata Dash, Anastasia Ailamaki: Automated physical design in database caches, Proceedings of ICDE Workshops (SMDB 2008), pp. 27-34, Cancun, Mexico, 2008.

[5] Umar Farooq Minhas, Jitendra Yadav, Ashraf Aboulnaga, Kenneth Salem: Database systems on virtual machines: How much do you lose?, Proceedings of ICDE Workshops (SMDB 2008), pp. 35-41, Cancun, Mexico, 2008.

[6] Sandeep Tata, Lin Qiao, Guy M. Lohman: On common tools for databases - The case for a client-based index advisor, Proceedings of ICDE Workshops (SMDB 2008), pp. 42-49, Cancun, Mexico, 2008.

[7] João Pedro Costa, Pedro Furtado: Poster session: Towards a QoS-aware DBMS, Proceedings of ICDE Workshops (SMDB 2008), pp. 50-55, Cancun, Mexico, 2008.

[8] Hannes Voigt, Wolfgang Lehner, Kenneth Salem: Poster session: Constrained dynamic physical database design, Proceedings of ICDE Workshops (SMDB 2008), pp. 63-70, Cancun, Mexico, 2008.

[9] Mohamed A. Sharaf, Shenoda Guirguis, Alexandros Labrinidis, Kirk Pruhs, Panos K. Chrysanthis: Poster session: ASETS: A self-managing transaction scheduler, Proceedings of ICDE Workshops (SMDB 2008), pp. 56-62, Cancun, Mexico, 2008.

[10] Haider Rizvi, Joyce Coleman, Sam Lightstone: Poster session: Simplifying business intelligence with a hybrid appliance: the IBM balanced warehouse, Proceedings of ICDE Workshops (SMDB 2008), pp. 71-78, Cancun, Mexico, 2008.