Research Statement

Stratos Papadomanolakis

My research is motivated by the challenge of managing massive data. As diverse scientific fields such as astronomy and computational mechanics accumulate observational or experimental data, large-scale data management is expected to be the "limiting or the enabling factor for a wide range of sciences" [3]. The same holds for modern enterprises, which rely on massive data warehouses for business intelligence.

My broad research interests span all aspects of large-scale data management performance, such as query execution and optimization, indexing, storage system design and support for novel applications. I focus on two key research areas with a critical impact on large-scale data management. The first is *self-tuning* databases and in particular algorithms for automated database design. Automated design is crucial for performance and a challenging task given today's complex systems and workloads. The second is efficient indexing, query execution and disk layouts for datasets such as computational meshes, graphs and arrays, that are common in scientific and enterprise computing, but not effectively handled by existing databases.

1 Self-Tuning Databases

1.1 Workload-Driven Schema Partitioning

In large-scale databases, query performance depends on the design of structures such as tables, indexes and materialized views. The downside of using indexes and materialized views to improve performance, is the cost of storing them and the overhead of maintaining them during data updates.

I developed AutoPart [12], a novel workload-driven tool that improves query performance *without requiring additional storage or maintenance*. AutoPart combines horizontal (row-wise) and vertical (column-wise) table partitioning to reduce I/O costs for each query by eliminating unnecessary accesses to non-relevant data. AutoPart is novel also in that it is the first workload-driven partitioning tool implemented and evaluated in the context of modern commercial systems, with real-world database applications and workloads. I evaluated AutoPart using the Sloan Digital Sky Survey (SDSS) [5], a popular online astronomy database, whose multi-terabyte contents are hosted by a commercial database management system. Using a workload consisting of representative queries captured in the SDSS usage logs, I demonstrated that partitioning combined with a small set of key indexes outperforms index-only designs by up to 20% for queries and up to five times for updates, while requiring half the space for indexes.

The original paper on AutoPart [12] was invited for publication in an IEEE Data Engineering Bulletin issue on database support for the sciences [6] as one of the best papers in SSDBM'04. The latest application of AutoPart is in a DBMS-based mid-tier caching system developed at Johns Hopkins University and used by the *SkyQuery* astronomy database federation [13].

1.2 New Models and Algorithms for Database Design

State-of-the-art database design tools exhibit two main limitations. First, they perform a large number of optimizer calls in order to obtain accurate estimates for query costs and compare the benefits of alternative designs. While the query optimizer accurately models query execution costs, it is computationally expensive: without the optimizer overhead, design algorithms would execute orders of magnitude faster. Second, they use ad-hoc, application-specific search heuristics, that make it difficult to characterize their effectiveness in terms of how close they get to the optimal solution.

My research introduces a new framework for query cost estimation that improves the performance of database design tools by orders of magnitude. My framework is based on caching the result of a small number of key optimizer invocations, performed during a pre-computation phase. During the

normal algorithm execution, the cached query plans are *reused* to very efficiently and accurately obtain query costs without further optimizer invocations. I essentially introduce a novel flavor of *Parametric Query Optimization*, applicable to database design problems. The *INdex Usage Model* (INUM) [9] is an application on index selection, designed and evaluated with commercial database systems and optimizers. I showed that INUM provides cost estimates three orders of magnitude faster compared to optimizer-based approaches and showed that existing, commercial index selection tools can readily take advantage of the INUM's performance benefits.

I address the lack of formal analysis in design algorithms through the innovative application of combinatorial optimization techniques. I introduced a novel Integer Linear Programming (ILP) formulation and evaluated its benefits in the context of index selection algorithms. Unlike previous research, I was able to apply combinatorial optimization techniques to database design and obtain useful new results, such as a method for deriving non-trivial optimality bounds for approximate solutions. My approach differs from previous theoretical models by incorporating important aspects such as optimizer-based cost estimation, leading to practical ILP-based implementations. My preliminary results with an ILP formulation for index selection demonstrate that it allows for faster and better algorithms and were well received by the self-tuning database community [7].

2 Support for multidimensional data access

2.1 Query processing on unstructured tetrahedral meshes

Computational sciences today pose new data management challenges, that are not sufficiently addressed by the database community. For instance, the finite element meshes used in the Quake Project at Carnegie Mellon for earthquake modeling [2] consume hundreds of gigabytes, while finite element simulation results are even larger and therefore difficult to analyze. While previously it was feasible to fit meshes in main memory, the current data volumes require disk-based processing.

My research introduces data management for *unstructured tetrahedral meshes*, a data organization central to scientific computing. Tetrahedral meshes differ from the regular, rectangular-grid meshes used in previous work, by comprising pyramid-shaped elements (tetrahedra) with varying sizes and angles. Tetrahedral meshes are capable of modeling complex geometries, such as an earthquake ridge, but their geometry makes it difficult to build effective indexes.

I developed Directed Local Search (DLS) [8], a novel indexing and query processing algorithm for tetrahedral meshes that processes spatial queries such as points and ranges. The novelty of my approach is in the use of mesh topology information, essentially the graph describing the "connectivity" of mesh elements, for spatial query processing. Due to the use of mesh topology, DLS is insensitive to mesh geometry and outperforms previous work relying on geometric approximations (such as minimum bounding rectangles). Using datasets from several real-world applications, I demonstrated that DLS is up to four times faster compared to the existing best-performing multidimensional indexing schemes.

Furthermore, DLS is based on the ubiquitous B-Tree, does not require the implementation of new exotic access methods and can be easily integrated in existing databases. DLS is currently being implemented within SQL Server 2005, as part of a data management application for fracture mechanics, developed by Gerd Heber from the Cornell Fracture Group and Jim Gray from Microsoft Research [4].

2.2 Efficient disk I/O on multidimensional data

Multidimensional query performance depends on the efficiency of retrieving data from the disk. Databases try to preserve *locality* when arranging data on the disk, so that queries result in large I/O requests to contiguous disk blocks, that take advantage of the sequential disk bandwidth. Such an arrangement is not straightforward for multidimensional data, because it requires "fitting" multiple logical dataset dimensions into the single "dimension" implied by the contiguous physical arrangement of disk blocks.

My research revisits the traditional "linear disk" assumption and proposes MultiMap, a new disk model suitable for efficient multidimensional data access. MultiMap arranges disk blocks into a logical n-dimensional grid so that accesses along any dimension are efficient. MultiMap builds on a new abstraction for disk access that relies on groups of non-contiguous disk blocks that can be efficiently retrieved as a sequence, with short seek times and zero rotational latency.

My experimental results demonstrate the superiority of MultiMap compared to previously proposed layout techniques based on space-filling curves. MultiMap was well-received in both the storage [10] and database [11] communities. The paper on the storage aspects of MultiMap [10] won a *best paper award* (selected over twenty five accepted papers).

3 Future Work

3.1 Self-tuning databases

I am very interested to continue working in the area of self-tuning databases, because it combines huge commercial impact with challenging optimization problems. My existing work addresses key issues related to the performance and quality of algorithms for automated database design. In the future, I plan to work on the challenge of building *adaptive* database design algorithms. Adaptivity is a crucial feature, given the variability in both the query workloads and the database contents.

The first problem I will consider is the lack of a good characterization of change in SQL workloads. A characterization is important for both algorithm design and system design, for example to determine the frequency of actual database modifications. In collaboration with JHU researchers, I plan to conduct my study using the extensive activity logs collected by the SkyQuery distributed astronomy database federation [13].

As a first application for adaptive design, I will pursue the integration of AutoPart (Section 1.1) with the caching framework developed by JHU for SkyQuery. AutoPart will optimize query performance inside the SkyQuery mid-tier "caches", which in reality are full-fledged database systems storing portion of the federation data. The mid-tier cache is a natural target for adaptive design, due to the highly dynamic nature of both query workloads and the cache contents.

Another tough problem for automated design algorithms is the "integrated" design of multiple database structures. Exploring the simultaneous use of indexes, materialized views and partitioning clearly has the highest possible impact on performance, but causes an exponential increase in the number of design alternatives that must be considered. I intend to take advantage of the scalability offered by the INdex Usage Model (INUM) and my Integer Linear Programming (ILP) formulation to develop fast algorithms for "integrated" database design that reach solutions with measurable quality and incorporate them in a new generation of database design tools.

3.2 Database support for multidimensional scientific data

I will continue working towards the big goal of extending database technology to efficiently support scientific data management. I intend to extend the ideas behind my Directed Local Search (DLS) technique, to provide support for other scientific data types. An important problem is querying *time-varying* meshes, whose structure changes in each time-step of a simulation. Such datasets grow very large in size, as they might contain meshes for thousands of timesteps. I will look into extending DLS to handle structural changes to the mesh over time and obtain solutions that share the ease of implementation and performance of the original technique.

A key challenge is indexing high-dimensional point sets, a general problem with applications in almost every scientific domain. The advantage of DLS-style query processing for high-dimensional points is that it can be easily integrated in existing database systems, thus increasing the reach of database technology towards a broad range of applications. The approach for enabling DLS in this setting is to add structure to the data by triangulating a subset of the input points. Using the resulting space partitioning, DLS will be used to efficiently process point, range and nearest-neighbor queries.

Finally, as most scientific datasets today are stored in files (using formats like HDF [1]), it is important to extend databases with scientific file access functionality. The integration will allow scientists to take advantage of the advanced metadata management capabilities offered by the database without sacrificing the high-performance raw data access that is a typical feature for files. Among the interesting research questions raised, is finding the best overall database design that includes both files and table-based storage and incorporating external file access methods in query processing and optimization. I am collaborating with Gerd Heber from the Cornell Theory Center, in a project that also involves Microsoft engineers interested in adding scientific file access capabilities to the SQL Server database.

References

[1] http://www.hdfgroup.org/.

[2] www.cs.cmu.edu/ quake.

[3] Office of science data-management workshops: The office of science data-management challenge. Technical report, Department of Energy, 2005.

[4] G. Heber and J. Gray. Supporting finite element analysis with a relational database backend part ii: Databa se design and access. Technical Report MSR-TR-2006-21, Microsoft Research, 2006.

[5] J.Gray, D.Slutz, A.Szalay, A.Thakar, J.vandenBerg, P.Kunszt, and C.Stoughton. Data mining the SDSS skyserver database. Technical Report MSR-TR-2002-01, Microsoft Research, 2002.

[6] S. Papadomanolakis and A. Ailamaki. Workload-driven schema design for large scientific databases. *IEEE Data Eng. Bull.*, 2004.

[7] S. Papadomanolakis and A. Ailamaki. An integer linear programming approach to database design. In *SMDB '07, ICDE Workshop on Self-Managing Databases*, 2007.

[8] S. Papadomanolakis, A. Ailamaki, J. C. Lopez, T. Tu, D. R. O'Hallaron, and G. Heber. Efficient query processing on unstructured tetrahedral meshes. In *SIGMOD '06: Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, 2006.

[9] S. Papadomanolakis, D. Dash, and A. Ailamaki. Intelligent use of the query optimizer in automated database design. Technical Report CMU-CS-06-151, Computer Science Department, Carnegie Mellon University, 2006.

[10] S. W. Schlosser, J. Schindler, S. Papadomanolakis, M. Shao, A. Ailamaki, C. Faloutsos, and G. R. Ganger. On multidimensional data and modern disks. In *Proceedings of the 4th USENIX Conference on File and Storage Technology (FAST '05)*.

[11] M. Shao, S. W. Schlosser, S. Papadomanolakis, J. Schindler, A. Ailamaki, and G. R. Ganger. Multimap: Preserving disk locality for multidimensional datasets. In *IEEE 23rd International Conference on Data Engineering (ICDE 2007)*.

[12] S.Papadomanolakis and A.Ailamaki. AutoPart: Automated schema design for large scientific databases using data partitioning. In *Proceedings of the 16th International Conference on Scientific and Statistical Database Management*, 2004.

[13] X. Wang, T. Malik, R. Burns, S. Papadomanolakis, and A. Ailamaki. A workload-driven unit of cache replacement for mid-tier database caching. In *DASFAA '07, 12th International Conference on Database Systems for Advanced Applications*, 2007.