# Fast Computation of Database Operations using Graphics Processors

Department of Computer Science University of North Carolina at Chapel Hill November 2004

## The Challenge

We present new algorithms for performing fast computation of several common database operations on commodity graphics processors. Specifically, we consider operations such as conjunctive selections, aggregations, and semi-linear queries, which are essential computational components of typical database, data warehousing, and data mining applications. Moreover, these operations are widely used as fundamental primitives to build complex database queries and to support on-line analytic processing (OLAP) and data mining procedures. The efficiency of these operations has a significant impact on the performance of a database system.

While graphics processing units (GPUs) have been designed for fast display of geometric primitives, we utilize the inherent pipelining and parallelism, single instruction and multiple data (SIMD) capabilities, and vector processing functionality of GPUs, for evaluating boolean predicate combinations and semi-linear queries on attributes and executing database operations efficiently.

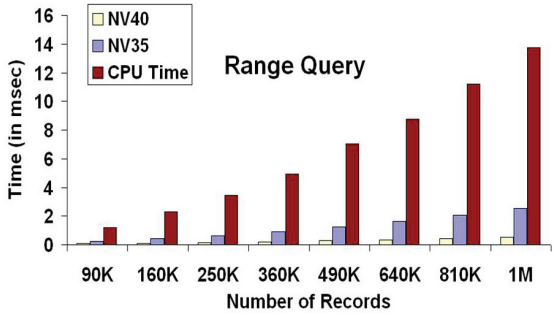
Our algorithms take into account some of the limitations of the programming model of current GPUs and perform no data rearrangements. We present novel algorithms for performing multiattribute comparisons, semi-linear queries, range queries, computing the kth largest number, and other aggregates. The attributes in the database are represented using floating point textures on current GPUs. The database queries are performed on these attributes by copying the data values from the textures into the depth buffer using fragment programs. The queries are then evaluated by using the depth test functionality of GPUs, and the results are stored in the stencil buffer.

These algorithms have been applied to large databases composed of up to a million records. The performance of these algorithms depends on the instruction sets available for fragment programs, the number of fragment processors, and the underlying clock rate of the GPU. We also perform a preliminary comparison between GPU-based algorithms running on a NVIDIA GeForceFX 5900 Ultra graphics processor and optimized CPU-based algorithms running on dual 2.8 GHz Intel Xeon processors.

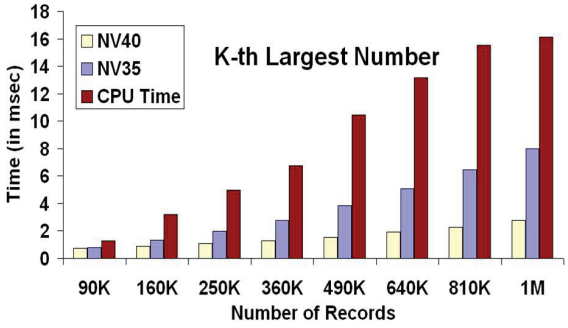
## Highlights

**We have developed novel algorithms for fast computation of several common database operations using graphics processing units (GPUs). Specifically, we consider essential computational operations for database and data mining applications such as:**

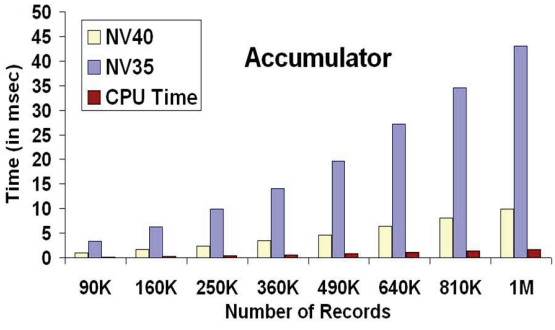
- **Conjunctive selections**
- **Aggregations**
- **Semi-linear queries**
- **Selectivity Analysis**




**High Performance Gain:** For semi-linear and selection queries, our GPU-based algorithms are 20 – 90 times faster than CPU-based implementations.




**Medium Performance Gain:** The algorithms for aggregates obtain modest gain of 6 – 9 times speedup over CPU-based implementations.




**Low Performance Gain:** Accumulator algorithm is slower than CPU-based implementation due to the lack of integer arithmetic on GPUs.

Our results can be summarized as follows:

• **High Performance Gain:** Our results (shown in Fig. 1 and 2) indicate that the semi-linear and selection queries map very well to GPUs and we are able to obtain significant performance improvement over CPU-based implementations.

• **Medium Performance Gain:** The algorithms for aggregates obtain a modest gain of 2 - 4 times speedup over CPU-based implementations.

• **Low Performance Gain:** In some cases, we did not observe any gain over a CPU-based implementation. Our GPU based ACCUMULATOR algorithm is slower than the CPU-based implementation. The slow performance is due to the lack of integer arithmetic instructions on current GPUs and slower clock as compared to CPUs.

Several new features are desirable for improving the functionality and performance of our algorithms.

• **Precision:** Current GPUs have depth buffers with a maximum of 24 bits. This limited precision can be an issue. With the increasing use of GPUs in performing scientific computing, graphics hardware developers may add support for higher precision depth buffers.

• **Integer Arithmetic Instructions:** Current GPUs do not offer integer arithmetic instructions in the pixel processing engines. In addition to database operations, several image and video compression algorithms also require the use of integer arithmetic operations. Given that the fragment programs were just introduced in the last few years, the instruction sets for these programs are presently being enhanced.

• **Depth Compare Masking:** Current GPUs support a Boolean depth mask that enables or disables writes to a depth buffer. It is very useful to have a comparison mask specified for the depth function, to that specified in the stencil function. Such a mask would make it easier to test if a number has i-th bit set.

• **No Random Writes:** The GPUs do not support random access writes, which makes it harder to develop algorithms on GPUs because they cannot use data rearrangement on GPUs.

Overall, our results indicate that the GPU can be used as an effective co-processor for many database operations.


### Team Members
**Naga Govindaraju**, research assisant professor
**Ming C. Lin**, professor
**Dinesh Manocha**, professor
**Wei Wang**, assistant professor
**Brandon Lloyd**, graduate student



### Research Sponsors
U.S. Army Research Office
Defense Advanced Research Projects Agency
Intel Corporation
National Science Foundation
Office of Naval Research


### Selected Publications

Naga K. Govindaraju, Brandon Lloyd, Wei Wang, Ming Lin, and Dinesh Manocha, Fast Computation of Database Operations using Graphics Processors, *Proc. of SIGMOD*, 2004.

Naga K. Govindaraju, Brandon Lloyd, Wei Wang, Ming Lin, and Dinesh Manocha, Fast Computation of Database Operations using Graphics Processors, Proc. of ACM Workshop on General Purpose Computing on Graphics Processors 2004, p. C-9



### Key Words
Query optimization, graphics processor


### For More Information
http://gamma.cs.unc.edu/DB