Whi te paper

“It is not science to build fast software on over-scaled hardware, but to build fast software on available resources”,

Peter Čapkovič, CTO of Instarea

Introducing TellStoryDB

Along with enormous increase in data volume, IT companies seek new possibilities of effective data processing. Especially, queries with real-time or near real-time responses in context of big data are super expensive or hardly possible to achieve with standard database systems and sometimes even various specialized noSql solutions.

TellStoryDB has utilized new hardware opportunities and it has integrated modern graphic cards that enable fast and efficient processing of vast amount of data. And therefore, even billions of records could be queried in milliseconds.

Analytical Specialization

In Instarea we work with big telco data what makes us challenged everyday with optimal approaches to manipulation with massive datasets. This project started as a reaction to our needs for fast querying over long tables filled with event records. So in the first stage, the main focus of TellStoryDB is to query a single flat and huge table in unprecedented fast and effective way. This approach will find its place in use-cases linked to repetitive querying over e.g. telco events data (counting number of sim cards in certain area with certain criteria), web logs (computing funnel analysis of certain website visitors), finance data, etc.

Basic Concepts of Superb Performance

TellStory uses two concepts that love each other and together they bring superb performance.

Utilization of Modern Hardware

Since 2007, when NVIDIA launched the CUDA Toolkit, using graphic processing units (GPUs) for high performance and scientific computing has become increasingly popular. This moved focus of graphics cards vendors from visualization towards more general computing. General processing graphics processing units (GPGPUs) enable highly parallelized processing of data thanks to hundreds and even thousands of cores doing work in single GPU. Although these cores can handle simpler logic than those of CPUs, their high number in single unit creates a powerful army. And in result it can process large data vectors much faster than any CPU.

Rediscovery of Columnar Data Storage

Utilizing GPUs computation power requires different approach to storing data. The most suitable database architecture that works well with parallel processing is columnar storage. In contrast to conventional relational databases which store data in row-based format, columnar databases store data in separate columns. In context of parallel processing, GPUs love long vectors of the same data type

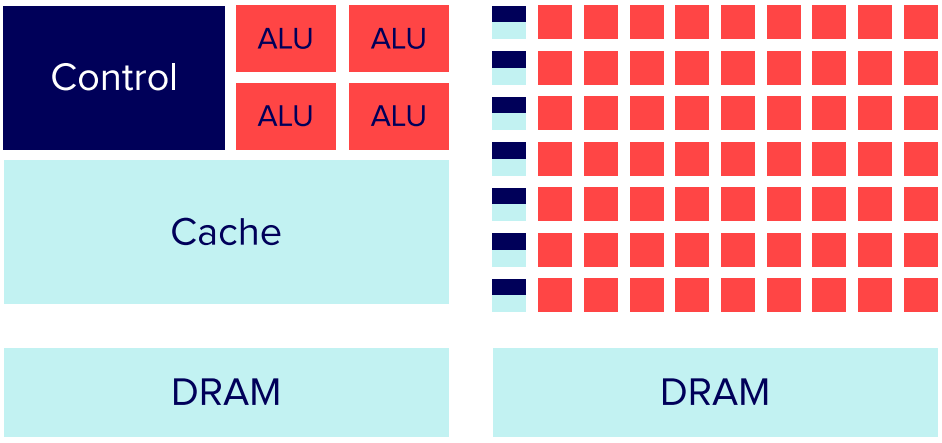

FIgure 1: GPUs have thousands of arithmetic logic units (ALUs) in one piece of hardware.

This concept is decades of years old, but it did not find its place in relational databases that were heavily used until era of big data started. Columnar databases have become popular mostly for analytical workloads on big datasets.

as all values are next to each other. In addition to this, data transfers are significantly reduced as only relevant columns are loaded and used for processing. Also, column-oriented databases are more suitable for horizontal scaling, enabling use of low-cost hardware to work in terabytes of data.

Architectural Insight

Data Flow Within System

Datasets are compressed and persisted on disc storage, but also cached in both CPU memory and GPU memory to minimize latency. GPU processing needs data to be present in its local memory (GPU RAM) and this results in main limitation of GPU computing – moving data between CPU and GPU memory.

TellStoryDB reduces this transfer time using three strategies. Firstly, when a query is hit, TellStoryDB analyzes which columns are relevant and only those are copied to GPU RAM. Secondly, thanks to compression of data, much lower volume needs to be transferred. Finally, data are kept in GPU RAM as late as possible to avoid duplicate transfers.

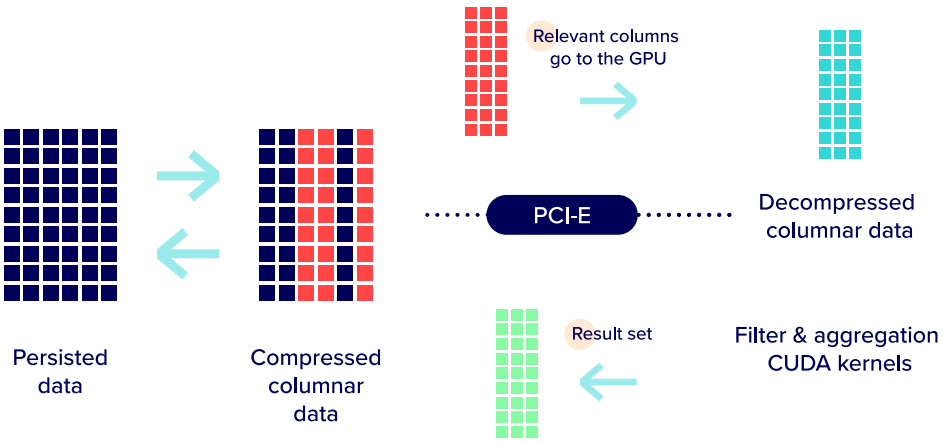

Figure 2: Data flow from disc storage, through main memory RAM to GPU RAM.

When data are transferred, GPU decompresses it and run functions on top of the data including various filters, aggregations, etc. At this point GPU can finally show how big beast it is.

GPU Caching

Especially, in analytics workloads it is very common to use one dataset more than once in a short period of time. Once data are transferred to GPU RAM it should remain there for repeated use. This is caching at GPU. Of course, GPU RAM has not unlimited capacity (usually ~16 GB) and we need to wisely manage memory when using various datasets. Thanks to caching, results could be expected in 10x shorter time when queried the same dataset.

Data Compression [Under Development]

As mentioned above, data are internally organized in columnar-wise format. One great advantage of columnar storage is that columns could be very efficiently compressed. TellStoryDB uses compression to reduce data volume on disk, but also across the whole hierarchy of memory. A compression scheme that is used is rather lightweight in order to provide maximum decompression speed. Therefore, data could be kept compressed even in GPU memory sparing space in GPU RAM that is naturally much smaller. And thus columns are decompressed only when necessary.

Software Toolkit Used

The database layer was implemented in C++ 17 and extension to this are CUDA kernels for computation at NVIDIA graphics cards. The Console is written in C# and therefor it requires .NET runtime environment,

Key Capabilities

SQL Language

In the world of databases, SQL is a common interface that almost every data developer or analyst knows and also many tools use it to communicate with persistence layer. TellStoryDB provides basic SQL statements to select and manipulate data.

Geospatial Functions

Based on our experience with telco data we added geospatial functionality to work with points and polygons. Currently, TellStoryDB supports inputs in WKT format and provides three main functions – CONTAINS (check whether point is inside polygon), INTERSECTION (new polygon as intersection of two polygons), UNION (new polygon as union of two polygons).

In-Memory

Data are preloaded into RAM at start. This way is latency decreased to minimum. Standard on-demand loading from disk is planned for next release in December 2019.

Indexing

TellStoryDB currently provides single clustered index. Similarly, to traditional databases, data are sorted according to chosen columns. Slower insert in turn results in magnitude of order faster querying.

Multiplatform

Big data world happens on Unix systems. We know that, but we also do not want to forget about companies that run Windows.

Scalability

Multiple GPUs within a single server node are detected automatically and the load is balanced also automatically. It is possible to use up to 8 Tesla GPUs in a single machine. Such machine is comparable to cluster of 20 high-end CPU nodes.

Connectors to Other Tools

In order to integrate the database into the processing pipeline we provide currently C# and C++ connectors and other connectors (including ODBC) are planned for next release in December 2019.