PhD Project of Marcell Fehér

Future communication networks and storage systems will face tremendous challenges to answer the increasing data traffic generated by end users, the novel requirements of 5G communications, and by the 50 to 500 billion sensor and actuation devices to be connected to the Internet of Things (IoT). Additionally, upcoming services and applications may also impose very low delay constraints on the transferred data. This will naturally prevent us from using current client-server infrastructures or popular cloud services. The main reason is that servers and cloud infrastructure are typically far away from the end-devices themselves, which translates into high delay to reach them. Furthermore, the costs of transferring massive amounts data all the way through the network to reach a cloud are prohibitive with 50 to 500 billion connected devices due to the economic costs in infrastructure capable of servicing these high loads and the energy costs for transferring to and processing at the cloud. Thus, developing technologies that (i) increase the throughput, (ii) reduce the delay, (iii) operate in a distributed fashion, (iv) are economically viable, and (v) store and process data close to the end devices, is crucial to the design and deployment of future communication networks and cloud computing.

This PhD project aims to help many of these challenges by using a new technique called Generalized Deduplication (GDD), which is a new compression technique developed at Aarhus University that has interesting properties for efficient data access. It can achieve substantial compression of data that has certain properties, ones that typical IoT data possesses. Due to the way GDD data is represented and the compression gains it can achieve, it's possible to realize serious advantages in multiple problems.

1. Having to store a fracture of the original data size increases the effective capacity of every storage device on the planet.
2. Transferring fewer bytes on the network results in faster communication and requires less bandwidth, allowing the current infrastructure to handle orders of magnitude more connected devices.
3. Compressing datasets in system memory and providing query capabilities directly on compressed data allows embedded and other low-memory end devices to perform analytics, moving part of processing from the centralized servers to the edge of the network. As a side effect, in-memory analytics

tools in desktop and cloud environments gain a huge boost in the size of dataset they are able to handle.

The following sections discuss in detail concrete research directions that we are planning to pursue during the PhD project.

Please note that we are going to use the term *compressed data* as a synonym of data that has gone through Generalized Deduplication.

Analytics on compressed data
The goal is to enable running different queries directly on compressed data. This allows us to significantly increase the size of datasets that can be analyzed with a given amount of system memory and storage.
The analyzed data can either be a structured format like CSV, JSON, columnar data like Apache Parquet, or unstructured like images, audio or video. We are aiming at supporting multiple types of queries.

- Basic queries on structured data (CSV, JSON, columnar formats): get rows of the given index or range of indices
- Basic queries on images: return a segment (rectangle) of the image
- Text search
- Rich queries on structured data: query by multiple columns, execute functions on the results (e.g. average, min, max, k-means)

We are aiming to provide optimizations for data access patterns of widely used AI algorithms, so they can work on top of compressed data without any modifications. Additionally, we are planning to provide a query language that new projects can use to query their data. We will look into the possibility to integrate our method into existing data analytics projects like Hadoop, Apache Spark and others.

We will also explore the possibility of providing quick approximate results to different query types. This would be possible due to how GDD represents compressed data internally. By running the queries only on the *bases*, it takes significantly less computation to produce an imprecise result of the query. Our research will focus on what query types can be supported by this method and whether it’s possible to determine and control the precision of the approximate results.

A reference implementation will demonstrate how these methods can be packaged into a software library that reads uncompressed data and provides an API to query it, using significantly less memory than the original data size.

Use AI to create efficient GDD transformation functions

Currently, GDD transformation functions (the algorithms that are used to separate uncompressed data to *bases* and *deviations*, and later reconstruct the original data) are hand picked from a small number of methods that are proven to work correctly, for example CRC, Hamming codes or Reed-Solomon error correcting codes. Different codes and coding parameters yield the best results for different input data and optimization goals. For example, maximum compression of IoT sensor readings is usually best by using Hamming (63,57) codes.

We are going to investigate how AI methods can be trained to automatically select or construct the best transformation function (coding method) for the given compression job. Our intuition is that each AI-produced transformation function will be suitable for a narrow set of data types, for example, x-ray images or pressure sensor readings, as well as specific usage, like storage (maximum compression) or analytics (the compressed representation must support a set of queries).

Improve training data quality for AI

Training AI requires a lot of data, which is often noisy. We are planning to investigate whether GDD can help the training process by increasing the quality of training data. Our thesis is that transforming the input data to a set of *bases* and *deviations* naturally separates the high quality information (*bases*) from the noise (*deviations*). We will experiment with different AI methods to see whether training with higher quality data generated by GDD results in better performing models. Additionally, we want to investigate what's the effect of using synthetic training data, which we produce by reversing the GDD process. By creating artificial *deviations* and applying them to the corresponding *bases*, we can generate data that's similar to the original training data. We will test how the addition of synthetic training data changes the performance of different AI models. This method has the potential to decrease the cost of collecting training data.