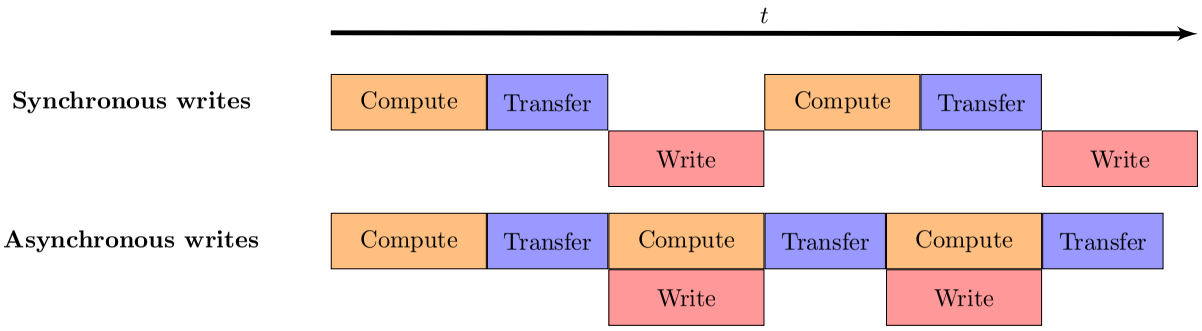

Figure 2.5: Asynchronous vs. synchronous communication (basic principle)

YakDB implements asynchronous writes using the ØMQ push/pull mechanism. While in the classical request/reply model, the client always expects a response message from the server, in the push/pull mode ØMQ only makes sure that the message is properly delivered over the network.

When using asynchronous communication, it needs to be ensured that *flow control* is properly enabled in the software. When the client constantly computes new datasets to write, the computation might be faster than the network or the processing of the data on the server side. Without flow control, excess data that can't be processed in time by subsequent algorithms will be temporarily stored in the main memory – at one point potentially exceeding the amount of memory available. Therefore, YakDB uses the ØMQ feature of queue watermarking in order to stop the client from generating new data when a predefined amount of data is already present in the queues. This mechanism is transparent to the programmer as in case of overflow the API will simply not return immediately but instead wait until the queue has free space available again.

Although implemented in many database systems, this approach is particularly advantageous for main-memory constrained systems as there are tight limits on how much memory can be consumed by the software at any given time.

Lightweight architecture

Due to its focus on resource-constrained devices, YakDB is built as a lightweight system, exhibiting the following core properties:

- No installation required: The minimal setup consist of a single executable and a configuration file in the same directory
- Configuration-free: No explicit configuration required, default configuration enables all features

- Fast startup: Less than 0.1 seconds[23] on a standard Linux computer
- Automatic table creation and setup: Once a table is used for the first time, it is automatically opened using the last table configuration or the configurable global default
- Tiny codebase of less than 5200 Source Lines of Code (SLOC)[24] plus less than 4200 SLOC Python interface code[25]

Although not all of those properties are relevant for most applications, it is clear that Translatron's approach of building a lightweight and thereby extensible is also reflected in YakDB. These properties are especially useful when extending the system. For a detailed discussion, refer to section 2.3.7.

Transparent compression

Although efficient algorithms like PERSIST (see section 2.4.3) significantly reduce the amount of storage space required for any particular setup of Translatron, this reduction is insufficient for many use cases.

Most of the storage space in a typical installation of Translatron being used to mine English texts is occupied by chunks of text and not binary data. According to [71] and [32] the overall entropy[26] of English text is low[27]. This is quite obvious as English has a very limited character set that is uniformly encoded using e. g. an 8-bit ASCII encoding[28] – therefore some of the eight bits are effectively wasted.

There is a multitude of compression algorithms available that can not only utilize the low entropy of the data stored in the database, but also recurring patterns, for instance common prefixes in a section of the database. The compression used by Translatron is transparent, meaning that the compression is not directly observed by the developer as the database is able to compress/decompress data on the fly and without manual interaction, thus reducing the total amount of disk space occupied by the Translatron database without loss of information.

Although the compression has a potentially negative impact on performance especially for random-access-heavy workloads (as in the worst case, any single access might require

[23] Maximum of 25 runs where a stop request was sent to the server immediately after startup.

[24] YakDB standalone server source lines of code, measured using sloccount 2.26.

[25] Includes inverted index and graph implementations, measured using sloccount 2.26.

[26] Information content in the context of computer science.

[27] As the exact entropy depends on the corpus in use and potentially other parameters, this discussion uses purely comparative terms instead of quantitative values. "low" is defined as entropy where less than 80% of the space occupied by a symbol is used for encoding information.

[28] For simplicity reasons, the discussion of variable-width encoding methods like UTF-8 including support for foreign language characters is avoided here.

a full block of compressed data to be decompressed), Translatron features selectable compression algorithms reaching from high-compression low-speed algorithms like bzip2 to real-time algorithms that can only save a small amount of disk space.

At the time of writing this thesis, the following compression algorithms are supported (see [23]):

- No compression
- bzip2
- Deflate
- Snappy
- LZ4
- LZ4HC

For heavily space-constrained devices it is also possible to extend RocksDB to support other algorithms: If space is more valuable than computing time, algorithms like LZMA or PAQ (see [58]) could be used that heavily compress the data and therefore save a significant amount of space when compared with faster algorithms. In principle, it is also possible to implement compression on a dedicated hardware platform to facilitate fast data processing even on low-end devices. However, this approach generally requires a significant effort and is commonly not cost-effective when compared to a solution where more powerful hardware like a notebook is used.

For text mining systems, asymmetrical compression algorithms are particularly suitable. These methods use more computing time for compression than for decompression and therefore are especially suited for applications where data is rarely written but frequently read.

2.3.4 Clustered architecture

Clustering in Excerbt

The old version of Excerbt, as published in [27], used only a single server with expensive and small[29], yet fast hard drives ([78, section 4.2.1]).

> *"These are very expensive disks (compared to regular commodity disks, such as SATA) and only available in limited storage capacities because of an exponential proportion of storage vs. price."* ([78, section 4.1])

[29] 1176 GiB in total, distributed over eight hard drives.