EZDB

Carter Brainerd

0xCB@protonmail.com

May 2018

1 Introduction

Popular databases like MySQL and PostgreSQL are, in many people's opinion, too complicated. At times, especially during installation, it seems like the developers are trying to make the process as difficult as possible. 90% of the bugs produced are usually for one of the two reasons:

1. Doing multiple things at one place
2. Doing one thing at multiple places[1]

Databases do too many things: manage users, plugins, web portals, and much more that *slow things down.* All these extraneous features decrease stability and security. What is needed is a *simple, secure, and speedy* database-like program to store data. The solution is a database that is stored in-memory and in a compiled language like C, C++, or Crystal. While Crystal is a relatively new programming language, it provides both the speed and type security as C as well as the simplicity of Ruby.

[1] Pravin Chaudhary. Twitter Post. https://twitter.com/pseudo_coder

2 Solution

EZDB is an in-memory key-value store used primarily for inter-process communication (IPC), but can have other uses as well (such as caching). EZDB uses local sockets for two or more client processes to transfer information to one another quickly. When starting the program, the user can define a max bytesize for each value.[2] By default the max size will be 128 bytes to keep the memory footprint to a minimum.[3]

3 Simplicity

EZDB at its heart is simple. As stated before, the main server program is written and compiled with the new Crystal language. Since EZDB is a key-value store, a complex query language like SQL is not needed. Instead, a small set of custom commands are used to get and set data. The commands will be discussed later. This simplicity allows EZDB's applications to be flexible. For example, if a developer wanted to use EZDB as a caching service, only the bytesize would need to change. Fetching and storing small bits of information in memory is immensely faster than from disk.[4]

4 Security

Due to its lack of complex machinery EZDB inherently has little to no security flaws. It never touches any system commands or passes any information directly to the kernel. Also, due to Crystal's static type checking,[5] value types are never changed or converted. Port 28468, the daemon port, is never opened to the internet, it is always bound to localhost, so there is no possibility of outside parties getting access.

---

[2] Note: This is not implemented yet (May 2018) but is in development.
[3] Ibid.
[4] Adam Jacobs. *Pathologies of Big Data*. https://queue.acm.org/detail.cfm?id=1563874
[5] https://crystal-lang.org/docs/syntax_and_semantics/types_and_methods.html

5 Speed

Since Crystal is compiled, it is inherently faster programs written in interpreted languages like Ruby or Python. The program itself is actually incredibly small (624 KB, ~100 lines of code at time of writing).[6] The small size of the program causes the memory footprint of it to be lower, leaving more memory for storing data.

All data in EZDB is transported over local sockets. UNIX socket support *may* be implemented in the future, but there are no solid plans to implement them yet.

There are no plans to implement compression on the server side. This is because with small chunks of data, the compressed data can actually be larger than the uncompressed data. However, if clients wish to use compression, they are welcome to do so (so long as they keep the compression consistent between the client programs).

To handle multiple requests at once, Crystal unique Fiber concurrency system. A fiber is in a way similar to an operating system thread except that it's much more lightweight and its execution is managed internally by the process.[7] The use of lightweight fibers helps increase the efficiency and therefore speed of EZDB.

6 Syntax

At the time of writing, EZDB has 4 distinct commands with a simple syntax:

6.1 set

The `set` command sets the value in the given key. If the command succeeds, the value is passed back over the socket.

Ex. `` `set key1 value1` `` would print `value1` to the socket.

[6] https://github.com/cbrnrd/ezdb/blob/master/src/ezdb/server.cr
[7] https://crystal-lang.org/docs/guides/concurrency.html

6.2 qset

The `qset` command does the same thing as the `set` command, but does not return the value if the command succeeds.

6.3 unset

The unset command sets the value at the given key to a blank string (“”).

6.4 get

The `get` command gets the value associated with the given key.

Ex. `` `get key1` `` would return value1 if that was the value of key1.

7 Implementations

As a proof of concept, a Java client library has been developed (ezdb4j - GitHub). I personally have plans to develop client libraries for Python, Ruby, Go, and C/C++ (likely in that order) sometime in the future.

8 Conclusion

We have proposed a program for storing small chunks of data, simply, securely, and quickly. The program is intended to be used for inter-process communication (IPC), but has a wide array of other uses as well. Two or more processes send data to the daemon over a local socket in order to share data between each other quickly.