Subject: Key/Value store disk-based caching solution
Posted by jjacksonRIAB on Wed, 11 Sep 2019 05:24:12 GMT
View Forum Message <> Reply to Message

I've already looked at lmdb and people have written wrappers for it in C++ but I'd rather have something that fits in better with how U++ works. As far as I can tell it uses a btree and COW with memory mapped files. I think it would be nice, and probably not a lot of work, to write a U++ style wrapper for it unless there are other options. I have an example discord client I'm working on to learn more about U++ and I'd like to be able to cache data like images, Values, etc because it becomes very expensive/slow to grab that data off the Internet each and every time. It would also be nice to have it mmapped so that I don't have to generate copies of that data and can reference it directly.

What would you recommend?

Subject: Re: Key/Value store disk-based caching solution
Posted by mirek on Wed, 11 Sep 2019 08:20:12 GMT
View Forum Message <> Reply to Message

jjacksonRIAB wrote on Wed, 11 September 2019 07:24I've already looked at lmdb and people have written wrappers for it in C++ but I'd rather have something that fits in better with how U++ works. As far as I can tell it uses a btree and COW with memory mapped files. I think it would be nice, and probably not a lot of work, to write a U++ style wrapper for it unless there are other options. I have an example discord client I'm working on to learn more about U++ and I'd like to be able to cache data like images, Values, etc because it becomes very expensive/slow to grab that data off the Internet each and every time. It would also be nice to have it mmapped so that I don't have to generate copies of that data and can reference it directly.

What would you recommend?

Wrapper would be nice.

Alternatives (from the head): To cache things from the internet (or from over the network), traditional approach to cache them as files works well. Done that in the past.

For key/value storage, if I would be too lazy to use other options (like creating a wrapper), I would use Sqlite3 or MySQL with trivial schema as key/value storage.

Mirek

Subject: Re: Key/Value store disk-based caching solution
Posted by jjacksonRIAB on Wed, 11 Sep 2019 09:42:30 GMT
View Forum Message <> Reply to Message

mirek wrote on Wed, 11 September 2019 10:20

Wrapper would be nice.

Alternatives (from the head): To cache things from the internet (or from over the network), traditional approach to cache them as files works well. Done that in the past.

For key/value storage, if I would be too lazy to use other options (like creating a wrapper), I would use Sqlite3 or MySQL with trivial schema as key/value storage.

Mirek

Yes, right now I'm caching images as files which works just fine, but there is also metadata associated with the images and text that would be very helpful to gather in a KVS. In discord each comment a person makes has a snowflake associated with it and you can start "replaying" missed content while you were offline by providing that snowflake, I believe, at which point it gives you every change made after it (edits, deletes, etc). Right now I'm doing it the "dumb" way and just asking for the last 100 posts but that is not only slow reloading every time you come back online but precludes people from going back even further in the history (pretty much everything in the channel's lifetime is saved in pages).

I think after I've gotten somewhere with it, I'll put it up on github so people can rip through it and tell me how the code can be improved. You've already helped me improve how I'm doing things quite a bit. I'll look into Sqlite3 or see if I can write an lmdb wrapper that has some method calls similar to U++ containers.

Subject: Re: Key/Value store disk-based caching solution
Posted by jjacksonRIAB on Thu, 10 Oct 2019 21:04:43 GMT
View Forum Message <> Reply to Message

I'm not completely happy with it but I decided I'd release early before I potentially go down the wrong path on this wrapper.

https://github.com/BornTactical/LmdbUpp

It doesn't even support a fraction of the things LMDB does and I'm sure I have bugs so I wouldn't touch anything in production with this, but it does work with Upp serialization. I'll have to iteratively refine it until it becomes something I'm happy with. Nothing is locked down, most everything is subject to change.
