Multi-Backend Caching Library in Go

Link: https://github.com/Devisree146/Go_project-library.git

Library set up:

go mod init <library>

github.com/Devisree146/Go_project-library

In main.go file import this library

Go mod init github.com/Devisree146/Go_project-library

Create a new Go_project, with a go.mod file and a go.sum file.

** go mod init—For go.mod file

** go mod tidy—For go.sum file

IN-MEMORY CACHE

These functions collectively manage an in-memory cache with LRU eviction policy, ensuring
thread-safe operations and periodic cleanup of expired items.

Methods for in_memory:

Set: Adds or updates an item in the cache. Moves updated items to the front of the LRU list
and evicts items if the capacity is exceeded.
Get: Retrieves an item from the cache if it exists and hasn't expired. Updates the item's
position in the LRU list.

Deletekey: Internally deletes an item from the cache without locking. Used in the eviction
routine and Get method.
GetAll: Retrieves all non-expired items from the cache. Initiates eviction for expired items.
Delete: Removes a specific item from the cache if it exists.
DeleteAll: Clears all items from the cache.
evict: Removes the least recently used (LRU) item from the cache when capacity is exceeded.

Test results of in_memory_cache

```
PS C:\Users\devis\OneDrive\Desktop\python-sample-tests\Go_project> go test ./test/in_memory_test -v
=== RUN   TestNewInMemoryCache
--- PASS: TestNewInMemoryCache (0.00s)
=== RUN   TestSetGet
--- PASS: TestSetGet (0.00s)
=== RUN   TestDelete
--- PASS: TestDelete (0.00s)
=== RUN   TestDeleteAll
--- PASS: TestDeleteAll (0.00s)
=== RUN   TestTTLExpiration
--- PASS: TestTTLExpiration (2.00s)
=== RUN   TestLRUEviction
--- PASS: TestLRUEviction (0.00s)
=== RUN   TestExists
--- PASS: TestExists (0.00s)
=== RUN   TestGetAllKeys
--- PASS: TestGetAllKeys (0.00s)
=== RUN   TestSetAndGet
--- PASS: TestSetAndGet (0.00s)
=== RUN   TestEvictionPolicy
--- PASS: TestEvictionPolicy (0.00s)
PASS
ok      unified/test/in_memory_test     2.172s
```

Benchmark results of in_memory_cache

```
PS C:\Users\devis\OneDrive\Desktop\python-sample-tests\Go_project\test\in_memory_test> go test -v -bench=. -ben
chmem ./test/in_memory_test/
=== RUN   TestNewInMemoryCache
--- PASS: TestNewInMemoryCache (0.00s)
=== RUN   TestSetGet
--- PASS: TestSetGet (0.00s)
=== RUN   TestDelete
--- PASS: TestDelete (0.00s)
=== RUN   TestDeleteAll
--- PASS: TestDeleteAll (0.00s)
=== RUN   TestTTLExpiration
--- PASS: TestTTLExpiration (2.00s)
=== RUN   TestLRUEviction
--- PASS: TestLRUEviction (0.00s)
=== RUN   TestExists
--- PASS: TestExists (0.00s)
=== RUN   TestGetAllKeys
--- PASS: TestGetAllKeys (0.00s)
=== RUN   TestSetAndGet
--- PASS: TestSetAndGet (0.00s)
=== RUN   TestEvictionPolicy
--- PASS: TestEvictionPolicy (0.00s)
PASS
ok      unified/test/in_memory_test     2.164s
```

REDIS CACHE

These functions collectively provide a Redis-based cache with LRU eviction policy and CRUD operations.

Methods for redis caching:
Set: Adds a key-value pair to the cache with a specified time-to-live (TTL). Updates the
access order for LRU and evicts elements if necessary.

Get: Retrieves the value for a given key from the cache and updates the access order for
LRU.
GetAll: Retrieves all key-value pairs currently stored in the cache.
Delete: Removes a specific key from the cache and the LRU list.
DeleteAll: Clears the entire cache by deleting all keys and the LRU list.

Test results of redis_cache

```
PS C:\Users\devis\OneDrive\Desktop\python-sample-tests\Go_project> go test ./test/redis_cache_test -v
=== RUN   TestRedisCache_SetGetDelete
--- PASS: TestRedisCache_SetGetDelete (0.02s)
=== RUN   TestRedisCache_GetAllKeys
--- PASS: TestRedisCache_GetAllKeys (4.14s)
=== RUN   TestRedisCache_DeleteAll
--- PASS: TestRedisCache_DeleteAll (4.16s)
PASS
ok      unified/test/redis_cache_test   8.528s
```

Benchmark results of redis_cache

```
PS C:\Users\devis\OneDrive\Desktop\python-sample-tests\Go_project\test\redis_cache_test> go test -v -bench=. -benchmem ./test/redis_cache_test/
=== RUN   TestRedisCache_SetGetDelete
--- PASS: TestRedisCache_SetGetDelete (0.01s)
=== RUN   TestRedisCache_GetAllKeys
--- PASS: TestRedisCache_GetAllKeys (4.12s)
=== RUN   TestRedisCache_DeleteAll
--- PASS: TestRedisCache_DeleteAll (4.14s)
PASS
ok      unified/test/redis_cache_test   8.468s
```

UNIFIED API:
Methods in Unified API:
SET: **POST /cache/**
Description: Sets a key-value pair in the cache with an expiration time.
Parameters: JSON object containing key, value, and expiration (in seconds).
Response: Returns a success message upon setting the value.

GET: **GET /cache ?key="Key value"**
Description: Retrieves the value associated with the specified key from the cache.
Parameters: key - The key to look up in the cache.
Response: Returns the value if the key is found; otherwise, returns a 404 error.

GETALL: **GET /cache/all**
Description: Retrieves all key-value pairs from the cache.
Response: Returns a JSON object containing all items in the cache.
DELETE: **DELETE /cache ?key="key value"**
Description: Deletes the specified key from the cache.

Parameters: key - The key to delete from the cache.
Response: Returns a success message if the key is deleted; otherwise, returns a 404 error.
DELETEALL: **DELETE /cache/all**
Description: Deletes all keys from the cache.
Response: Returns a success message if all keys are deleted; otherwise, returns a 404
error if no keys are found.
The API interaction can be done with Postman or using CURL commands as well.

Test results of multicache

```
PS C:\Users\devis\OneDrive\Desktop\python-sample-tests\Go_project> go test ./test/multicache_test -v
[GIN-debug] [WARNING] Creating an Engine instance with the Logger and Recovery middleware already attached.

[GIN-debug] [WARNING] Running in "debug" mode. Switch to "release" mode in production.
 - using env:   export GIN_MODE=release
 - using code:  gin.SetMode(gin.ReleaseMode)

[GIN-debug] POST   /cache/:key               --> unified/test/multicache_test_test.handleSet (3 handlers)
[GIN-debug] GET    /cache/:key               --> unified/test/multicache_test_test.handleGet (3 handlers)
[GIN-debug] DELETE /cache/:key               --> unified/test/multicache_test_test.handleDelete (3 handlers)
[GIN-debug] GET    /cache                    --> unified/test/multicache_test_test.handleGetAll (3 handlers)
[GIN-debug] DELETE /cache                    --> unified/test/multicache_test_test.handleDeleteAll (3 handlers)
=== RUN   TestCacheOperations
=== RUN   TestCacheOperations/TestSetAndGet
[GIN] 2024/07/09 - 10:33:15 | 200 |           0s |                 | POST     "/cache/testkey?value=testvalue"
[GIN] 2024/07/09 - 10:33:15 | 200 |           0s |                 | GET      "/cache/testkey"
=== RUN   TestCacheOperations/TestGetNonExistingKey
[GIN] 2024/07/09 - 10:33:15 | 404 |           0s |                 | GET      "/cache/nonexistent"
=== RUN   TestCacheOperations/TestDeleteNonExistingKey
[GIN] 2024/07/09 - 10:33:15 | 404 |           0s |                 | DELETE   "/cache/nonexistent"
--- PASS: TestCacheOperations (0.00s)
    --- PASS: TestCacheOperations/TestSetAndGet (0.00s)
    --- PASS: TestCacheOperations/TestGetNonExistingKey (0.00s)
    --- PASS: TestCacheOperations/TestDeleteNonExistingKey (0.00s)
PASS
ok      unified/test/multicache_test    (cached)
```

Benchmark results of multicache

```
PS C:\Users\devis\OneDrive\Desktop\python-sample-tests\Go_project> cd .\test\multicache_test\
PS C:\Users\devis\OneDrive\Desktop\python-sample-tests\Go_project\test\multicache_test> go test -v -bench=. -be
nchmem ./test/multicache_test/
[GIN-debug] [WARNING] Creating an Engine instance with the Logger and Recovery middleware already attached.

[GIN-debug] [WARNING] Running in "debug" mode. Switch to "release" mode in production.
 - using env:   export GIN_MODE=release
 - using code:  gin.SetMode(gin.ReleaseMode)

[GIN-debug] POST   /cache/:key               --> unified/test/multicache_test_test.handleSet (3 handlers)
[GIN-debug] GET    /cache/:key               --> unified/test/multicache_test_test.handleGet (3 handlers)
[GIN-debug] DELETE /cache/:key               --> unified/test/multicache_test_test.handleDelete (3 handlers)
[GIN-debug] GET    /cache                    --> unified/test/multicache_test_test.handleGetAll (3 handlers)
[GIN-debug] DELETE /cache                    --> unified/test/multicache_test_test.handleDeleteAll (3 handlers)
=== RUN   TestCacheOperations
=== RUN   TestCacheOperations/TestSetAndGet
[GIN] 2024/07/09 - 10:47:32 | 200 |            0s |                 | POST     "/cache/testkey?value=testvalue"
[GIN] 2024/07/09 - 10:47:32 | 200 |            0s |                 | GET      "/cache/testkey"
=== RUN   TestCacheOperations/TestGetNonExistingKey
[GIN] 2024/07/09 - 10:47:32 | 404 |            0s |                 | GET      "/cache/nonexistent"
=== RUN   TestCacheOperations/TestDeleteNonExistingKey
[GIN] 2024/07/09 - 10:47:32 | 404 |            0s |                 | DELETE   "/cache/nonexistent"
--- PASS: TestCacheOperations (0.00s)
    --- PASS: TestCacheOperations/TestSetAndGet (0.00s)
    --- PASS: TestCacheOperations/TestGetNonExistingKey (0.00s)
    --- PASS: TestCacheOperations/TestDeleteNonExistingKey (0.00s)
PASS
ok      unified/test/multicache_test    0.240s
```