WebKit Disk Cache

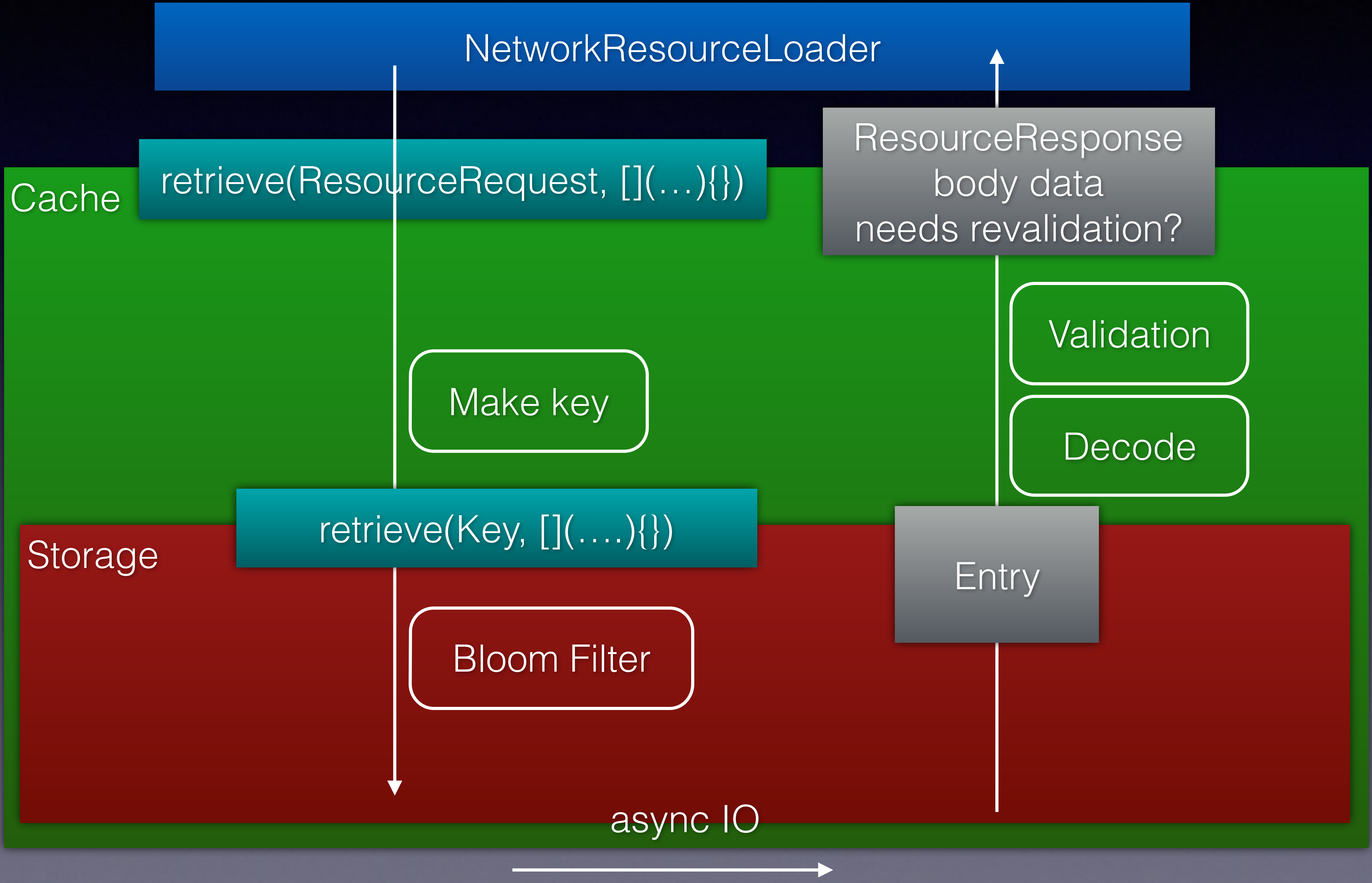
Why?

- Immediate performance gains
    - CPU
    - Latency
- Future innovation
- WebKit does caching already!

Design

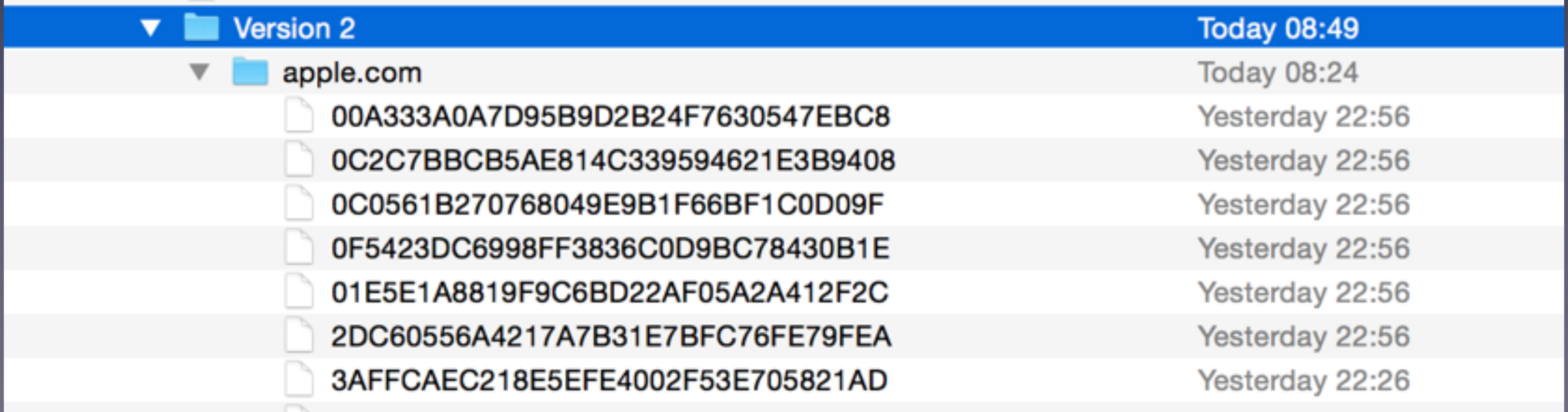
- Lives in the WebKit2 network process
- Zero global meta data (except file system!)
- One file per cache entry
- Async IO, WorkQueues
- Memory mapped data
- Modern C++

Retrieving Cache Entry



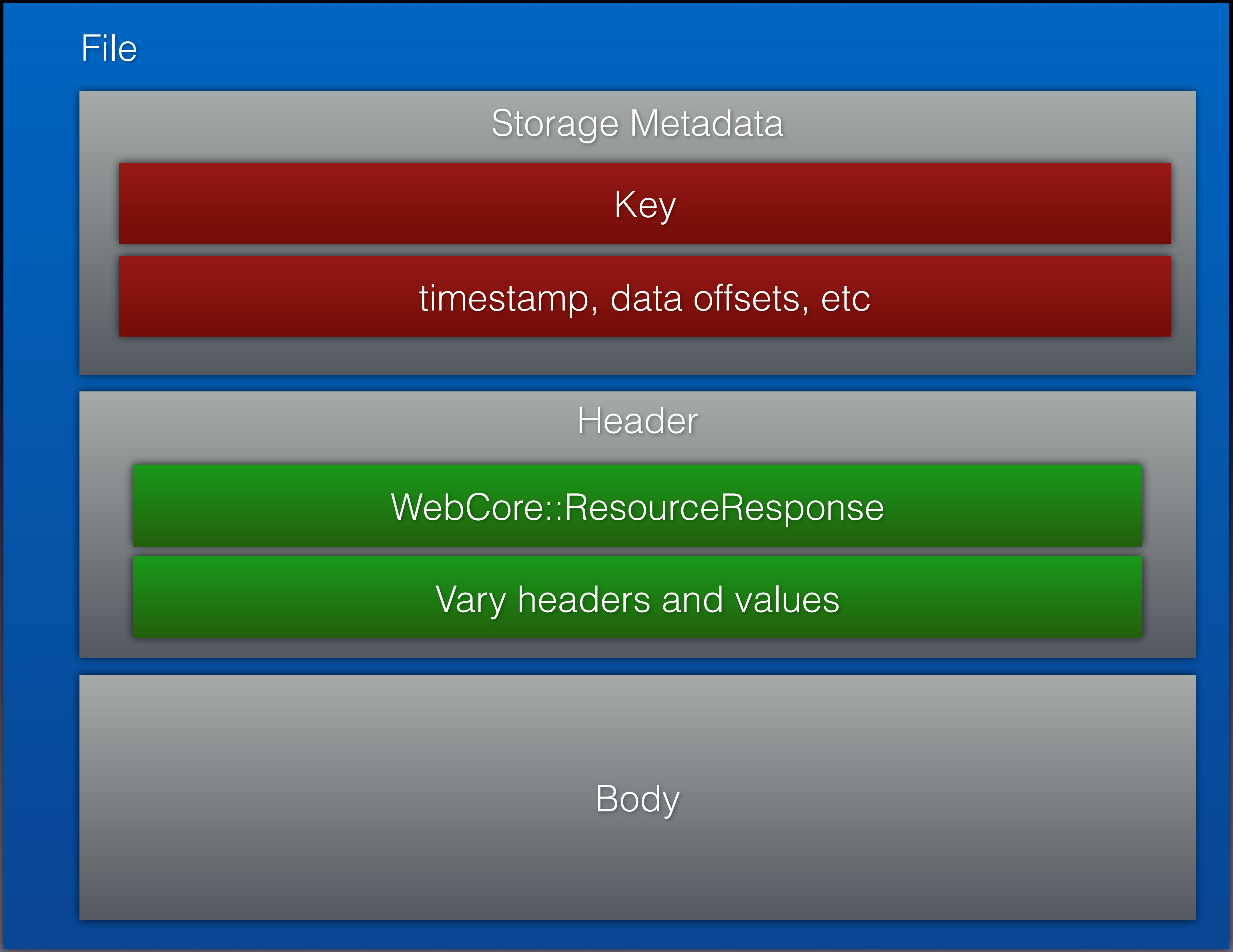
Cache Key

- Uniquely identifies cache entry
- Partition/request URL/method
- MD5 hash computed over key fields
- Hash used directly as file name for entries



▼ Version 2	Today 08:49
▼ apple.com	Today 08:24
00A333A0A7D95B9D2B24F7630547EBC8	Yesterday 22:56
0C2C7BBCB5AE814C339594621E3B9408	Yesterday 22:56
0C0561B270768049E9B1F66BF1C0D09F	Yesterday 22:56
0F5423DC6998FF3836C0D9BC78430B1E	Yesterday 22:56
01E5E1A8819F9C6BD22AF05A2A412F2C	Yesterday 22:56
2DC60556A4217A7B31E7BFC76FE79FEA	Yesterday 22:56
3AFFCAEC218E5EFE4002F53E705821AD	Yesterday 22:26

Structure of Cache Entry



File

Storage Metadata

Key

timestamp, data offsets, etc

Header

WebCore::ResourceResponse

Vary headers and values

Body

Porting

- Needs WebKit2 with network process
- Implementations for low level abstractions
    - WTF::WorkQueue
    - NetworkCache::IOChannel
    - NetworkCache::Data

Major components

- NetworkResourceLoader
  - Generates cache retrieves and stores, validates
- NetworkCache::Cache
  - API,HTTP cache logic and related
- NetworkCache::Key
- NetworkCache::Storage
  - Reading and writing bits to disk
- NetworkCache::Coders
  - Serialization/deserialization support
- NetworkCache::IOChannel, ::Data
  - Platform abstractions