CACHING STRATEGIES

BEN RAMSEY

HI, I'M BEN.

I'm a web craftsman, author, and speaker. I build a platform for professional photographers at ShootProof. I enjoy APIs, open source software, organizing user groups, good beer, and spending time with my family. Nashville, TN is my home.

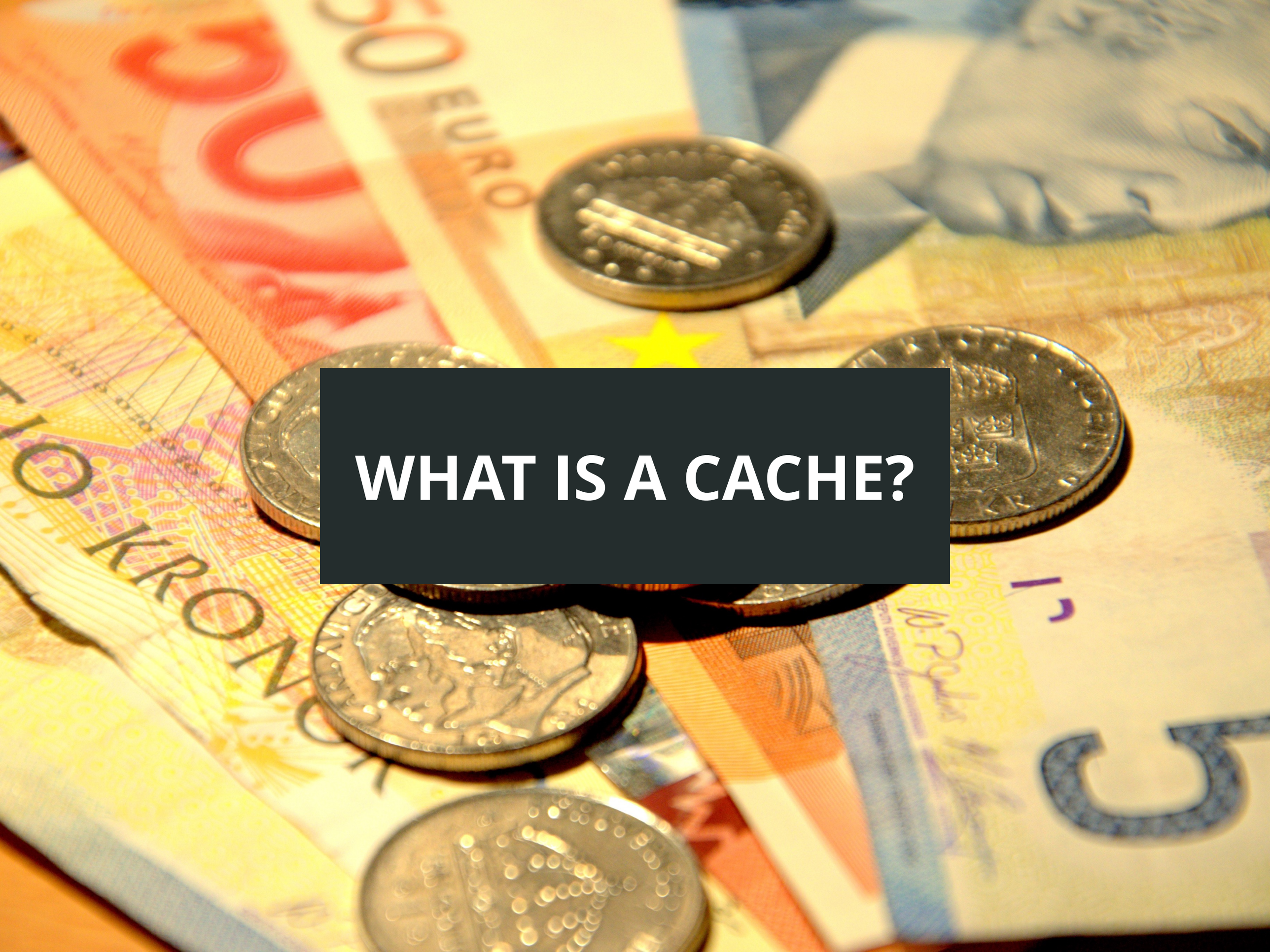
- Books
  - *Zend PHP Certification Study Guide*
  - *PHP 5 Unleashed*
- Nashville PHP & Atlanta PHP
- array_column()
- rhumsaa/uuid library
- virtPHP
- PHP League OAuth 2.0 Client

ShootProof [ ]

virtPHP

ShootProof

We're hiring.



WHAT IS A CACHE?

A CACHE IS...

A store of things that may be required in the future, which can be retrieved rapidly, protected, or hidden in some way.

- ✤ Animals store food in caches
- ✤ Journalists call a stockpile of hidden weapons a “weapons cache”
- ✤ Buried treasure is a cache
- ✤ Geocachers hunt for caches
- ✤ Computers and applications store data in caches

A CACHE IS…

A store of things that may be required in the future, which can be retrieved rapidly, protected, or hidden in some way.

IN COMPUTING, A CACHE IS...

A fast temporary storage where recently or frequently used information is stored to avoid having to reload it from a slower storage medium.

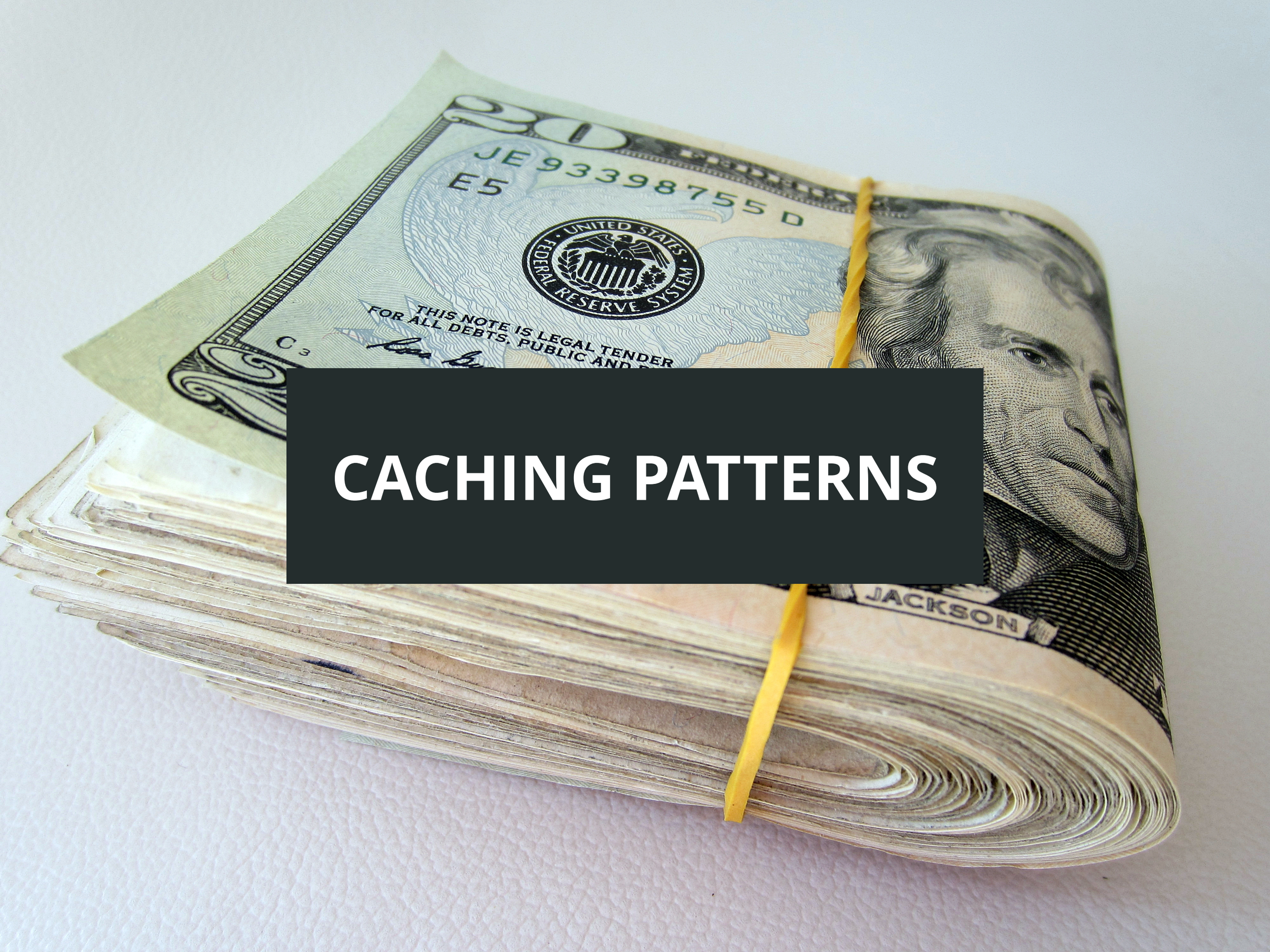
- ✤ Reduce the number of queries made to a database
- ✤ Reduce the number of requests made to services
- ✤ Reduce the time spent computing data
- ✤ Reduce filesystem access
- ✤ What else?

IN COMPUTING, A CACHE IS...

A fast temporary storage where recently or frequently used information is stored to avoid having to reload it from a slower storage medium.

OUR FOCUS...

Caching from the perspective of a web application.



CACHING PATTERNS

READ-THROUGH

If the item is not in the cache, the cache store requests the item from the data store and returns it, storing it in the cache.

- All reads go through the cache store
- If the cache store doesn't have the item, it requests it from the data store
- Functionality provided by the caching layer

WRITE-THROUGH

When updating items, update through the cache store, and it will propagate through to the data store synchronously.

- All writes go through the cache store
- Synchronous
- Operation not completed until it has written to the data store
- Functionality provided by the caching layer

WRITE-BEHIND

When updating items, update through the cache store, and it will propagate through to the data store asynchronously.

- All writes go through the cache store
- Asynchronous
- Data store updated in the background on a delay
- Functionality provided by the caching layer

REFRESH-AHEAD

When frequently-accessed objects in cache are near expiration, the cache store proactively refreshes the objects from the data store.

- Keeps the cache warm and fresh
- Reduced latency on cache lookups
- Functionality provided by the caching layer

CACHE-ASIDE

If the item is not in the cache, the application requests the item from the data store and stores it in the cache.

- Determine whether the item is in the cache
- If not in cache, read the item from the data store
- Store a copy of the item in the cache
- Emulate write-through by invalidating item in cache when updating data store
- Functionality provided by the application layer

TYPES OF CACHE

- File system
- Shared memory
- Object cache
- Database
- Opcode cache
- Web cache

FILESYSTEM CACHE

Perhaps the simplest way to
cache web application data:
store the generated data in local files.

CACHE HTML PAGES

Generate some HTML content, store it to a local file.

```
$html = '';

// Lots of code to build the
// HTML string or page.

file_put_contents(
    'cache.html',
    $html
);
```

CACHE HTML PAGES

Retrieve the pre-generated contents, if available.

```
$html = file_get_contents('cache.html')

if ($html === false) {
    $html = generateHtml();
    file_put_contents('cache.html', $html);
}

echo $html;
```

CACHE DATA STRUCTURES

Store populated data structures on the local filesystem.

```php
if (file_exists('cache.php')) {
  include 'cache.php';
}

if (!isset($largeArray)) {
    $largeArray = fooBuildData();

    $cache = "<?php\n\n";
    $cache .= '$largeArray = ';
    $cache .= var_export(
        $largeArray,
        true
    );
    $cache .= ";\n";

    file_put_contents(
        'cache.php',
        $cache
    );
}
```

CACHE.PHP

The created cache.php file now contains something that looks like this:

```
<?php

$largeArray = array (
  'db_name' => 'foo_database',
  'db_user' => 'my_username',
  'db_password' => 'my_password',
  'db_host' => 'localhost',
  'db_charset' => 'utf8',
);
```

/DEV/SHM

Many Linux systems these days automatically provide RAM disk mounted at /dev/shm. You may write to this in the same way you write to the filesystem, but it's all in memory.

```php
$configFile = '/dev/shm/config.php';

if (file_exists($configFile)) {
    include $configFile;
}

if (!isset($config)) {
    $config = getConfiguration();

    $cache = "<?php\n\n";
    $cache .= '$config = ';
    $cache .= var_export(
        $config,
        true
    );
    $cache .= ";\n";

    file_put_contents(
        $configFile,
        $cache
    );
}
```

OTHER APPROACHES

There are many other approaches to filesystem caching, but they’re all fundamentally the same.

- Store generated data to a file on disk.
- If available, read from that file on disk, rather than generating the data.
- If not available, generate the data and store it.
- That's how most caching works!

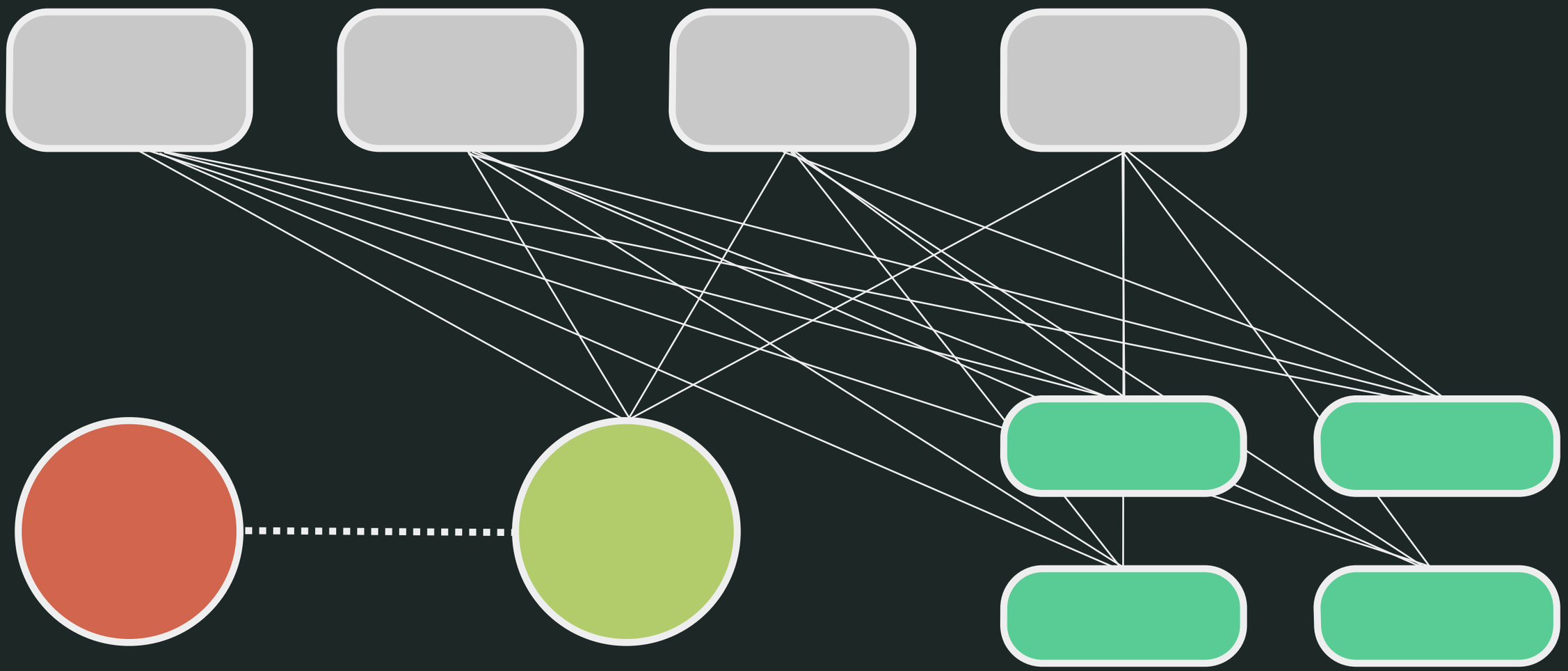
OBJECT CACHE

A variety of key-value arbitrary data stores exist.

MEMCACHED

Memcached is a distributed memory object caching system designed to store small chunks of arbitrary data.

- Simple key/value dictionary
- Runs as a daemon
- Everything is in memory
- Simple protocol for access over TCP and UDP
- Designed to run in a distributed pool of instances
- Instances are not aware of each other; client drivers manage the pool



PECL/MEMCACHED

Pecl/memcached is one of two PHP extensions for communicating with a pool of memcached servers.
pecl.php.net/package/memcached

```
$memcache = new Memcached();

$memcache->addServers([
    ['10.35.24.1', '11211'],
    ['10.35.24.2', '11211'],
    ['10.35.24.3', '11211'],
]);
```

GET AND SET WITH PECL/MEMCACHED

Use a key to set and retrieve data from a pool of memcached servers.

```
$book = $memcache->get('9780764596346');

if ($book === false) {
    if ($memcache->getResultCode() == Memcached::RES_NOTFOUND) {
        $book = Book::getByIsbn('9780764596346');
        $memcache->set($book->getIsbn(), $book);
    }
}
```

REDIS

Redis is another type of key-value data store, with some key differences.

- Supports strings and other data types:
  - Lists
  - Sets
  - Sorted sets
  - Hashes
- Persistence
- Replication (master-slave)
- Client-level clustering but built-in clustering in beta

PREDIS

Predis is perhaps the most popular and full-featured PHP client library for Redis. github.com/nrk/predis

```
$redis = new Predis\Client([
    'tcp://10.35.24.1:6379?alias=first-node',
    'tcp://10.35.24.2:6379?alias=second-node',
    'tcp://10.35.24.3:6379?alias=third-node',
]);
```

GET AND SET WITH PREDIS

In it’s simplest form, Predis behaves similar to the memcached client. However, it can perform complex operations, so check the docs.

```
$pageData = $redis->get('homePageData');

if (!$pageData) {
    if (!$redis->exists('homePageData')) {
        $pageData = getHomePageData();
        $redis->set('homePageData', $pageData);
    }
}
```

```php
$redis->hmset('car', [
    'make' => 'Honda',
    'model' => 'Civic',
    'year' => 2008,
    'license number' => 'PHP ROX',
    'years owned' => 1,
]);
```

```php
echo $redis->hget('car', 'license number');

$redis->hdel('car', 'license number');
$redis->hincrby('car', 'years owned', 1);
$redis->hset('car', 'year', 2010);

var_dump($redis->hgetall('car'));
```

DATABASE CACHE

Databases often have their own built-in caching mechanisms, and sometimes it’s useful to generate your own views.

QUERY CACHE

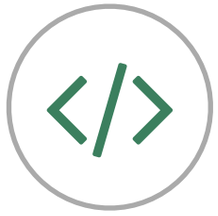
The query cache stores the SELECT statement together with the results. It returns these results for identical queries received later.

- Most database engines have something like this
- MySQL query cache no longer works for partitioned tables
- In a large, distributed application, is query-caching worth it? Or use something else, like memcached or Redis?

MATERIALIZED VIEWS

Sometimes queries with expensive joins need to be run beforehand, storing the results for later retrieval.

- Supported natively in Oracle and PostgreSQL
- Standard MySQL views do not solve this problem
- Triggers, stored procedures, and application code may be used to generate materialized views
- Simply a denormalized set of results, useful for fast queries



OPCODE CACHE

An opcode cache is a place to store precompiled script bytecode to eliminate the need to parse scripts on each request.

OPCACHE

The OPcache extension is bundled with PHP 5.5.0 and later. It is also available as an extension for PHP 5.2, 5.3, and 5.4. It is recommended over APC, which is similar.

php.net/opcache

```
// php.ini configuration

opcache.enable = "1"
opcache.memory_consumption = "64"
opcache.validate_timestamps = "0"
```

OPCACHE FUNCTIONS

OPCache comes with some useful functions that allow you to manage the scripts that have been cached.

```
opcache_compile_file($scriptPath)

opcache_get_configuration()

opcache_get_status()

opcache_invalidate($scriptPath)

opcache_reset()
```

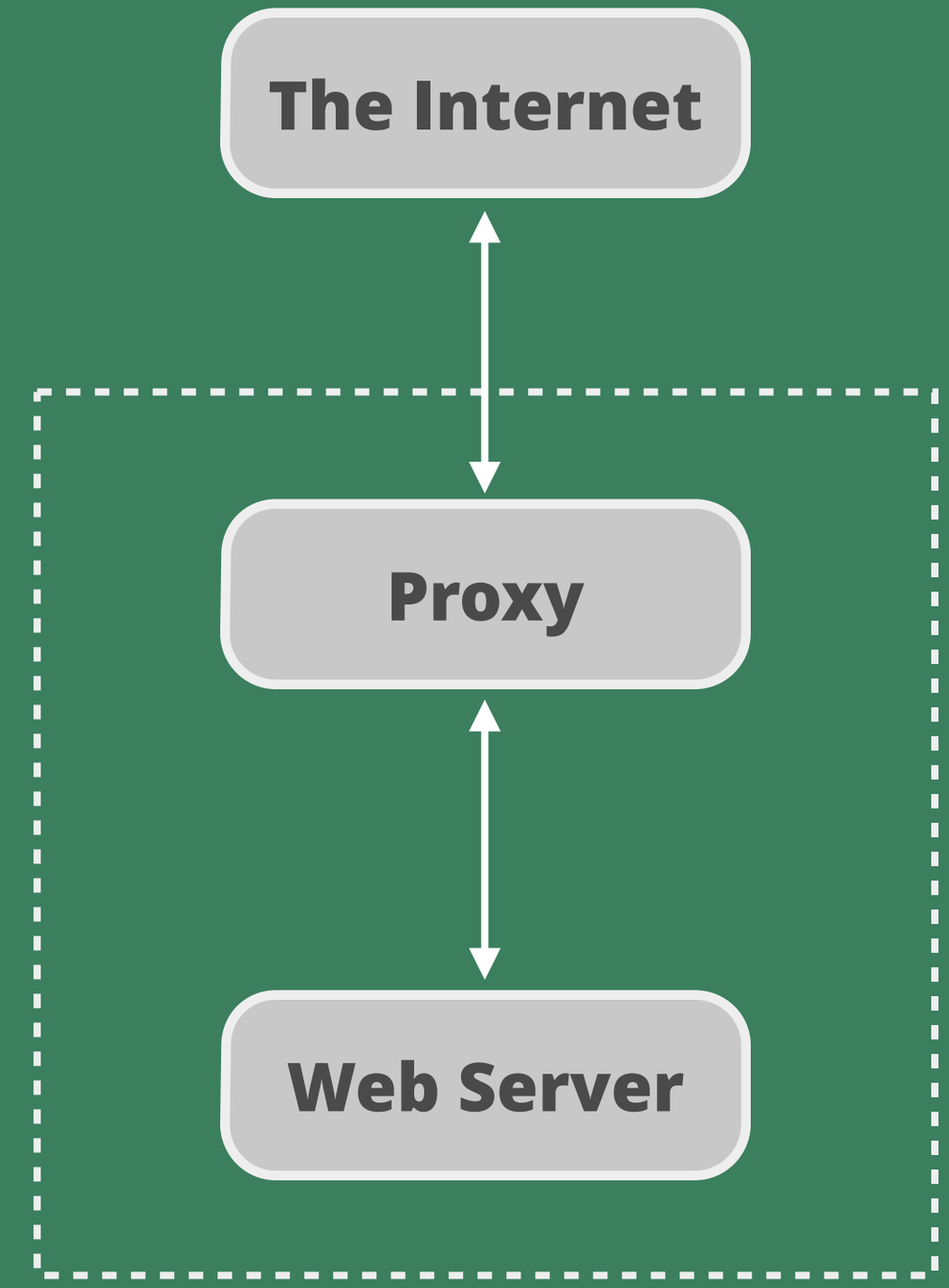
WEB CACHE

A web cache stores whole web objects, such as HTML pages, style sheets, JavaScript, and images.

REVERSE PROXY CACHE

A reverse proxy cache retrieves resources on behalf of a client from one or more servers and caches them at the proxy.

Sometimes called “web accelerators.”



EXAMPLES

There are many tools to help set up or use reverse proxy caches.

- Varnish Cache
- NGINX Content Caching
- Apache Traffic Server
- Squid
- Various CDNs provide this as part of their services

CONTENT DELIVERY NETWORK (CDN)

A CDN is a set of distributed servers in data centers across the globe with the purpose of delivering data from “edges” to speed up delivery to nearby users.

- Akamai Technologies
- Limelight Networks
- Level 3 Communications
- Amazon CloudFront
- Windows Azure CDN
- CloudFlare

GREENLAND
GREENLAND SEA
NORWEGIAN SEA
ICELAND
Reykjavik
Nuuk
LABRADOR SEA
Svalbard
BARENTS SEA
Moscow
UKRAINE
TURKEY
SYRIA
LEBANON
Al Jauf
Riyadh
SAUDI ARABIA
Cairo
EGYPT
Aswan
RED SEA
Mecca
LIBYA
Newfoundland
Halifax
Bermuda
Azores

HTTP CACHING

HTTP comes with a variety of headers for controlling freshness of requests.

- Expires
- Cache-Control
- Read Mark Nottingham's Caching Tutorial

CACHING TIPS

- Memoization
- Invalidation

MEMOIZATION

Technique used to store the results of expensive function calls and return the cached results when the same inputs occur again.

MEMOIZATION

For identical inputs, you always get the same output.

Hat tip to Larry Garfield for the code example.

```
function memoize($function) {
  return function() use ($function) {

    static $results = array();

    $args = func_get_args();
    $key = serialize($args);

    if (empty($results[$key])) {
      $results[$key] =
        call_user_func_array(
          $function,
          $args
        );
    }

    return $results[$key];
  };
}
```

MEMOIZATION

You can use this to wrap any callable and store/retrieve its results from the cache.

Hat tip to Larry Garfield for the code example.

```
$f = new Fancy();
$callable = [$f, 'compute'];
$f_cached = memoize($callable);

// And it really really works.
$f_cached($key);
```

INVALIDATION

Cache freshness is important, so we need ways to remove items from the cache or mark them as stale and invalid.

INVALIDATION

Keep your cache fresh.

- Set TTLs according to your needs
- Delete items (or update) items in the cache when items in the data store are updated
- Proactively review the cache and delete “stale” items
- Staleness and freshness are up to you

CACHE ALL THE THINGS!

APPENDIX: MYSQL QUERY CACHE NOTES

- Stewart Smith: Query cache removed from Drizzle because it doesn't scale on multi-core systems. Recommends deprecating it in MySQL.

- Rolando explains that query cache and InnoDB have been in a constant state of war, since InnoDB always inspects changes.

- Morgan Tocker: The query cache is off by default in MySQL 5.6 since it "does not scale with high-throughput workloads on multi-core machines. This is due to an internal global-lock, which can often be seen as a hotspot in performance_schema." Requests feedback from the community on use; his suspicion is that it is no longer needed.

THANK YOU.
ANY QUESTIONS?

If you want to talk more, feel free to contact me.

joind.in/13544

benramsey.com

@ramsey

github.com/ramsey

ben@benramsey.com

This presentation was created using Keynote. The design was inspired by the Catalyst web theme created by Pixelarity. The text is set in Open Sans. The source code is set in Fira Sans Mono. The iconography is provided by Font Awesome.

Unless otherwise noted, all photographs are used by permission under a Creative Commons license. Please refer to the Photo Credits slide for more information.

Caching Strategies
Copyright © 2015 Ben Ramsey

This work is licensed under Creative Commons Attribution-ShareAlike 4.0 International. For uses not covered under this license, please contact the author.

Ramsey, Ben. “Caching Strategies.” Lone Star PHP. Addison Conference Center, Addison, TX. 17 Apr. 2015. Conference presentation.

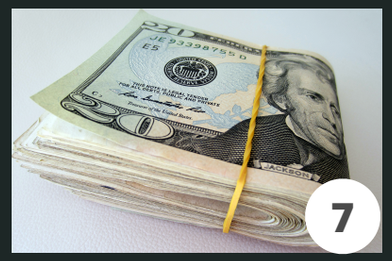


PHOTO CREDITS

1. “Lucky Loonie” by Sharon Drummond, CC BY-NC-SA 2.0
2. “Forex Money for Exchange in Currency Bank” by epSos.de, CC BY 2.0
3. “Cash Register” by Steve Snodgrass, CC BY 2.0
4. “Euro Note Currency” by www.TheEnvironmentalBlog.org, CC BY-NC-ND 2.0
5. “Various Currencies” by Bradley Wells, CC BY-NC-SA 2.0
6. “Riddle No. 5 — The Globe” by Graham, CC BY-NC-SA 2.0
7. “A Pile of Cash” by 401kcalculator.org, CC BY-SA 2.0