Caching Demystified

presented by

Aaron Welch

and

What is a cache?

A cache is a mechanism that allows you to store data that can be re-used later.

Common aspects of cache mechanisms include:

- store non-persistent data (will be destroyed at some point)
- depending on the type of cache, data can be cached as it is passed through the cache mechanism or stored and retrieved on request within an application
- provide some sort of interface to manually destroy all or part of the cached data

When are caches used?

- Many repeated operations
- Expensive operations

Why are caches used?

Volume

(support more requests, eg: people or traffic)

Performance

(go faster)

These things are not mutually exclusive; working to improve one, you generally get improvements on the other as well

Opcode Caches

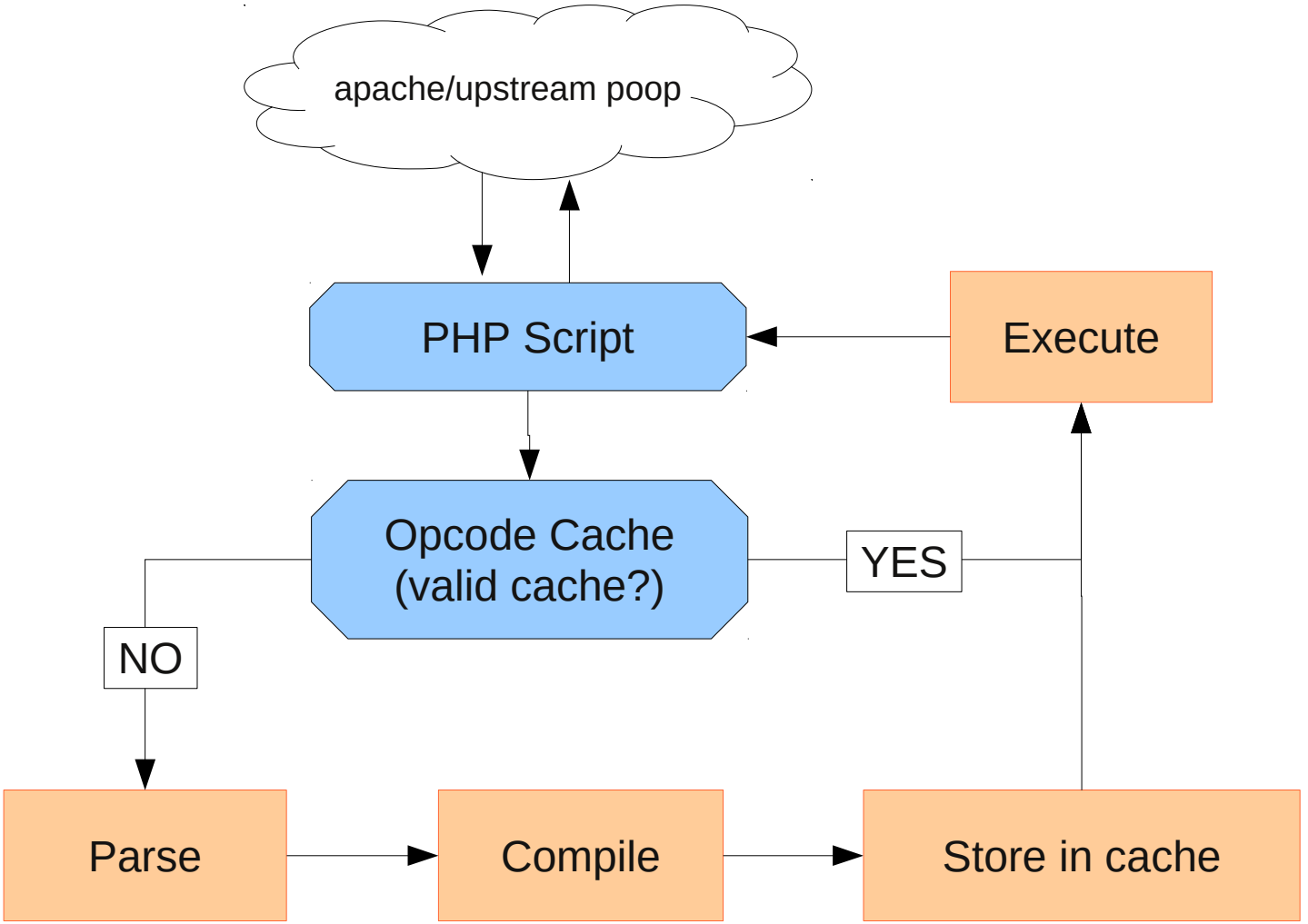
Caches PHP after being compiled, then re-uses that instead of compiling on every request.

Bonuses: Generally solutions are drop-in (high benefit with low integration cost), generally provide other PHP optimizations

Drawbacks: Does not generally help with application level problems like poorly written mysql queries or large memory footprint, does not help with page rendering speed in the browser, does not directly help with scaling volume in apache, only caches the php scripts – not anything generated by them

Examples: APC / eAccelerator / Zend Optimizer

Opcode Caches



Proxy Caches

Stores content delivered by the application (Drupal/Apache) and serves subsequent requests from cache instead of passing the request back to the application.

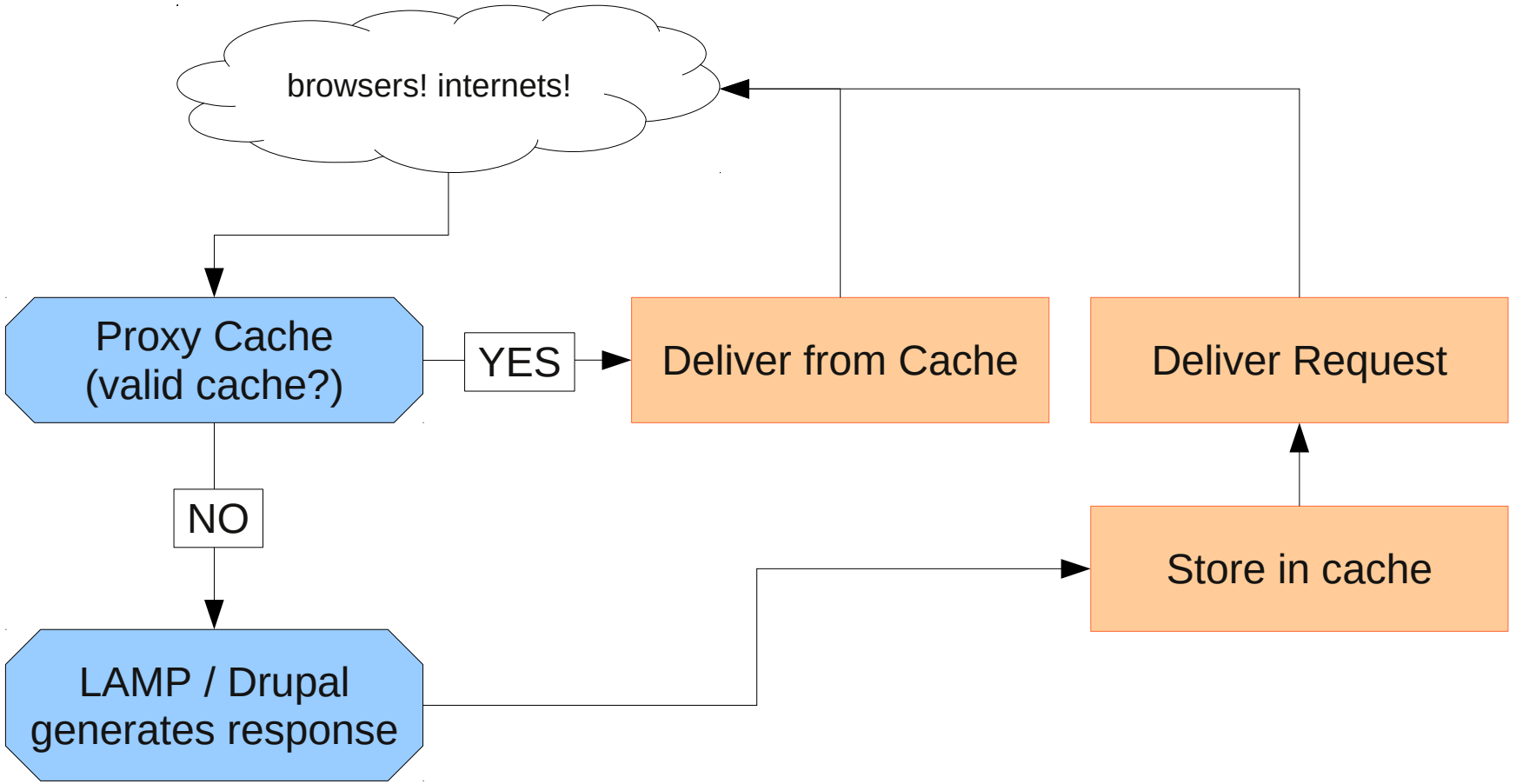
Bonuses: Removes the thread limitations of apache (scaling) and reduces load from Drupal / LAMP stack (performance), very high benefit, low resource requirement compared to Drupal alone

Drawbacks: Moderate to high integration cost, can make development/content management more difficult/confusing for developers/users, only effective for unauthenticated users

Tools: www.isvarnishworking.com to verify varnish

Examples: Varnish / Ngnix / Squid

Proxy Caches



Object Caches

Stores an object in memory by name, and returns that object on request later (name/value object store)

Bonuses: Extraordinarily performant, can drastically reduce database load for high transactional volume operations on tables with non-persistent data (sessions, cache tables)

Drawbacks: High integration cost, can make development/content management more difficult/confusing for developers/users, can be insecure if not protected (no auth mechanism), only helps with in-application bottlenecks (does not reduce memory footprint of the application itself)

Examples: memcache / membase / redis

Edge Caches

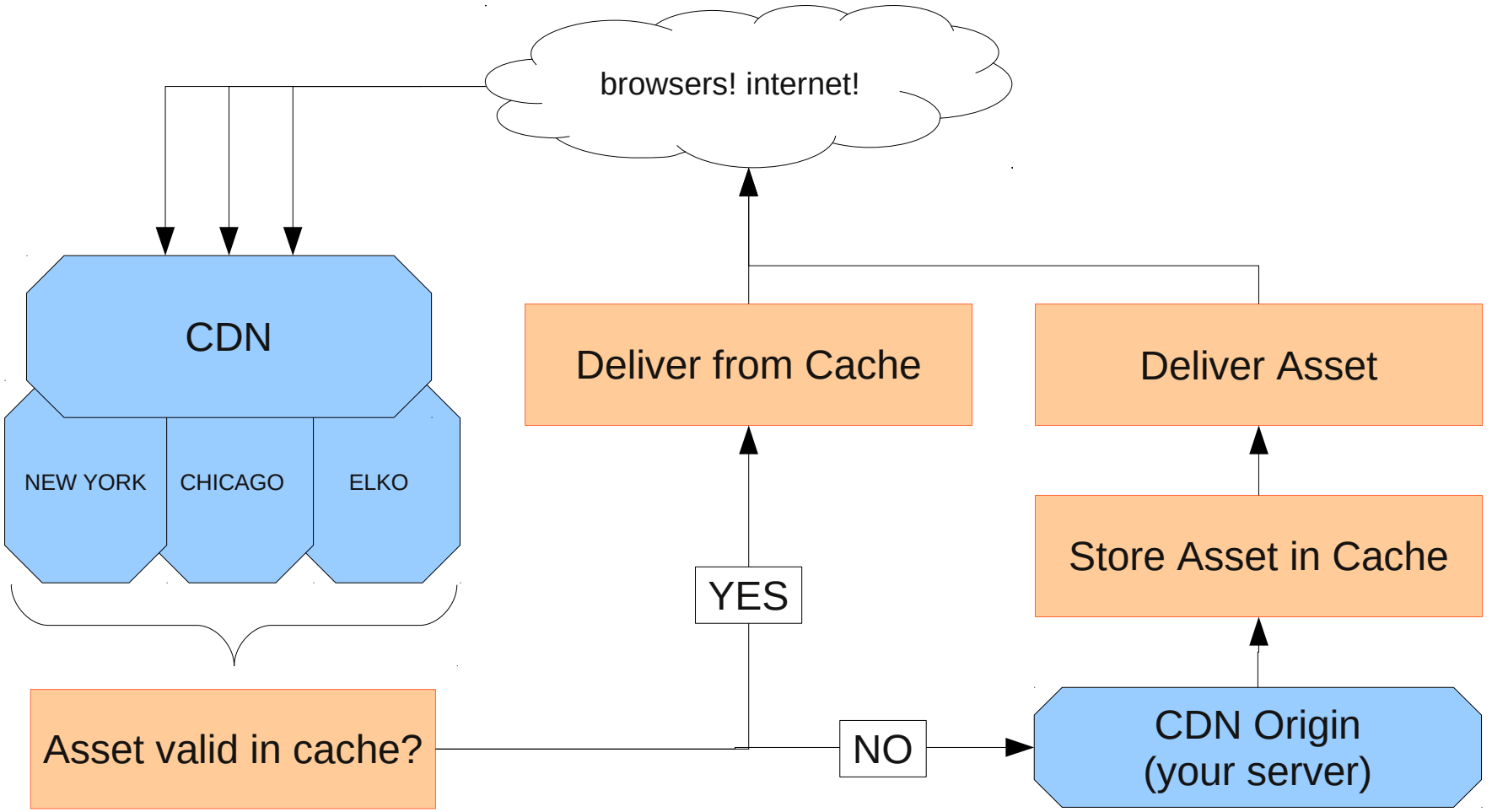
A network of servers that store requested assets in cache, and deliver them in a geographically relevant way

Bonuses: Massively scalable, reduces the number of requests for static assets on apache, can increase page rendering speed in the browser, easy to setup (not a local service)

Drawbacks: Not effective for low-volume sites (assets expire before enough requests are served from cache to be effective), can be problematic to integrate with the application, can make development/content management more difficult/confusing for developers/users

Examples: Akamai / Amazon cloudfront / cadre CDN (voxcast)

Edge Caches


browsers! internet!
CDN
NEW YORK
CHICAGO
ELKO
Deliver from Cache
Deliver Asset
Store Asset in Cache
YES
NO
Asset valid in cache?
CDN Origin
(your server)

Others

Browser
Database Server (internal)
Boost
DNS
Gold/Silver/Jewels

Questions?

www.getcadre.com

use coupon code BADCAMP2011 for one month free hosting