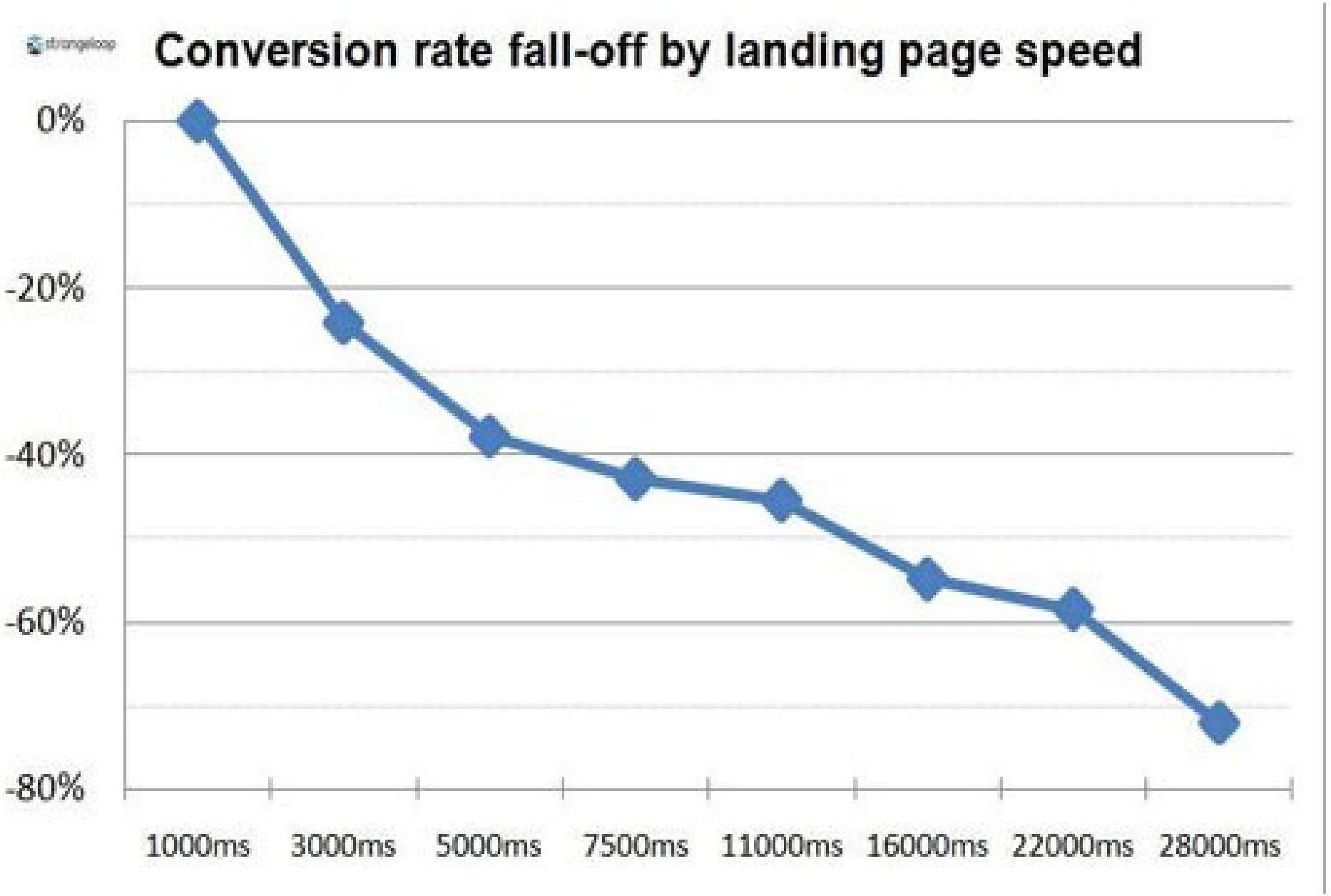
Cache Me If You Can

Dan Wilson + Mike Brunt + Mike Allen

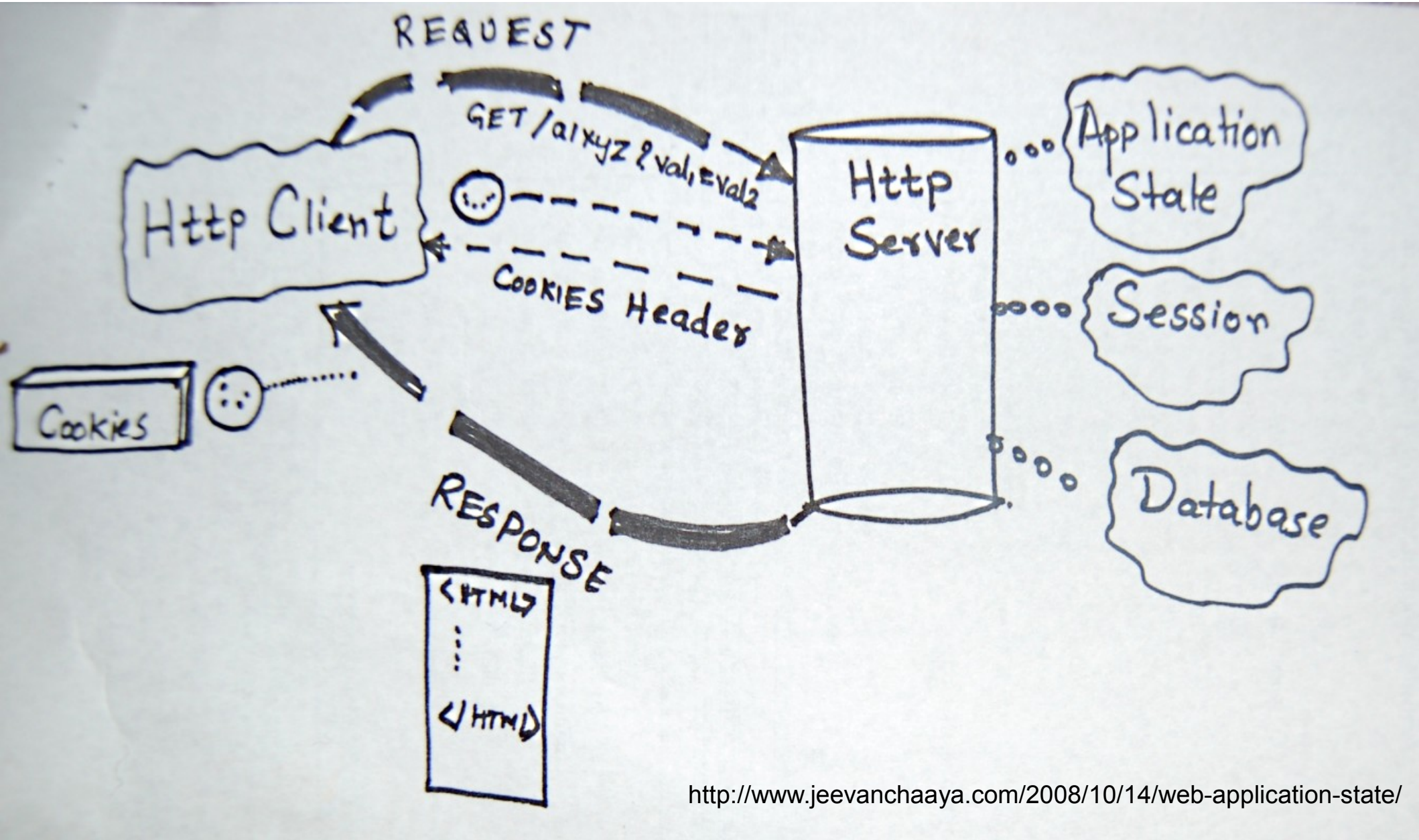
Why Caching is important


ST.ANTONY'S

Shared Resources = Wait Times


Conversion rate fall-off by landing page speed
0%
-20%
-40%
-60%
-80%
1000ms
3000ms
5000ms
7500ms
11000ms
16000ms
22000ms
28000ms

HTTP Request/Response



http://www.jeevanchaaya.com/2008/10/14/web-application-state/

Case Study: Wikimedia

Who is Wikimedia?

Wikipedia

Wiktionary

Wikinews

Wikibooks

Wikiquote

Wikisource

Wikiversity

Wikispecies

Wikimedia

- PHP/MySql
- 8 million articles over hundreds of language projects
- 110 million revisions
- 10th busiest site in the world (source: Alexa)
- 30 000 HTTP requests/s during peak-time
- 3 Gbit/s of data traffic
- 350 servers
- managed by ~ 6 people
- Wow

Could ColdFusion Power Wikimedia?


YUP

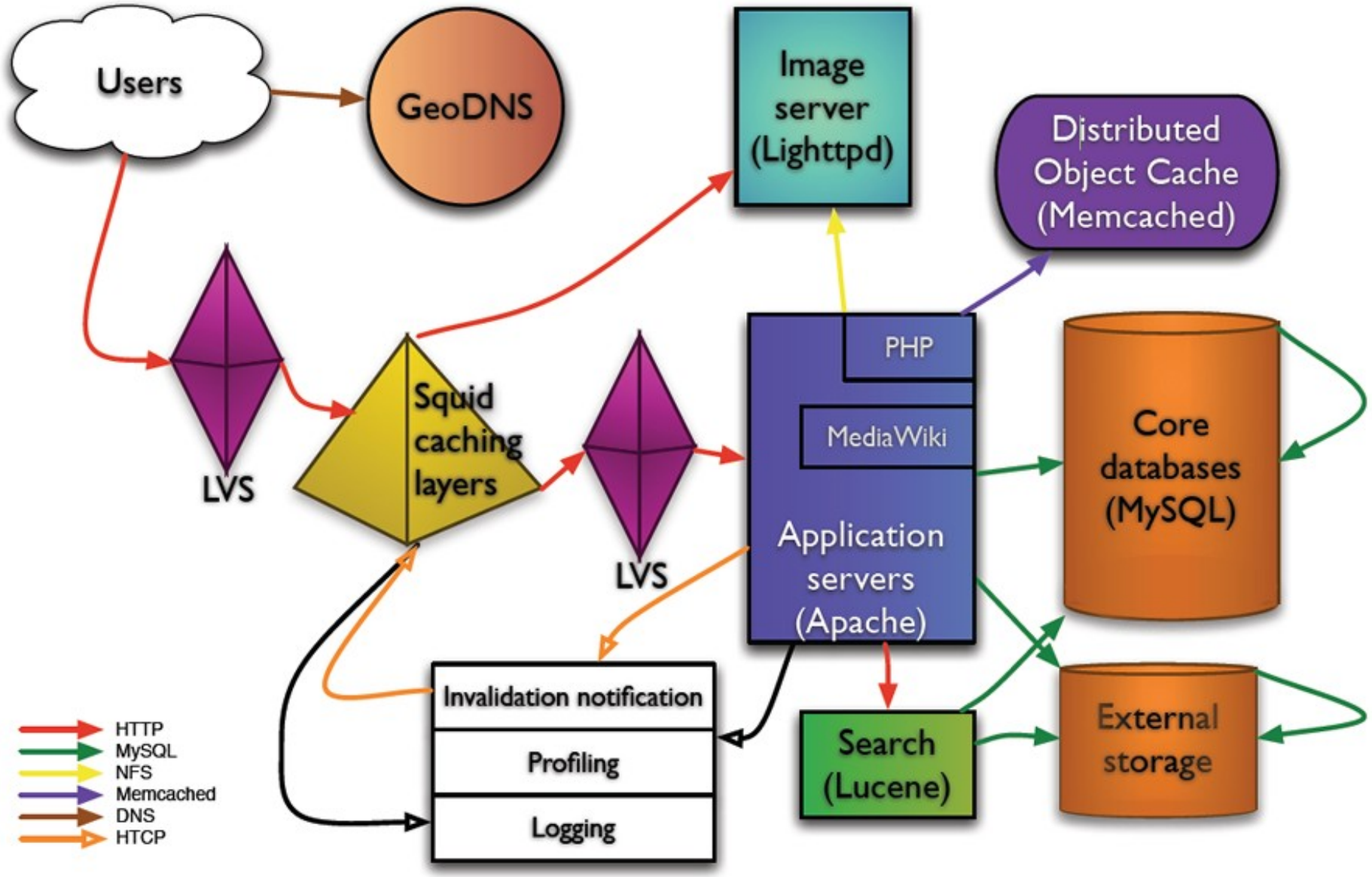
Wikipedia Functionality

Facts

- Users view Articles
- Articles have Revisions
- Revisions are immutable
- Articles must show latest revision
- Service must be able to handle surges (elections, world events, etc)

Questions

- What is an article?
- What is a Revision?
- When is the latest version needed?
- What is a surge?



Mark Bergsma, mark@wikimedia.org, Wikimedia Foundation Inc.

Wikimedia architecture

Reverse Proxy Cache (Front Cache)

Key Value Store

Key:

GET /ga.js HTTP/1.1

Host: www.google-analytics.com

User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US;

...

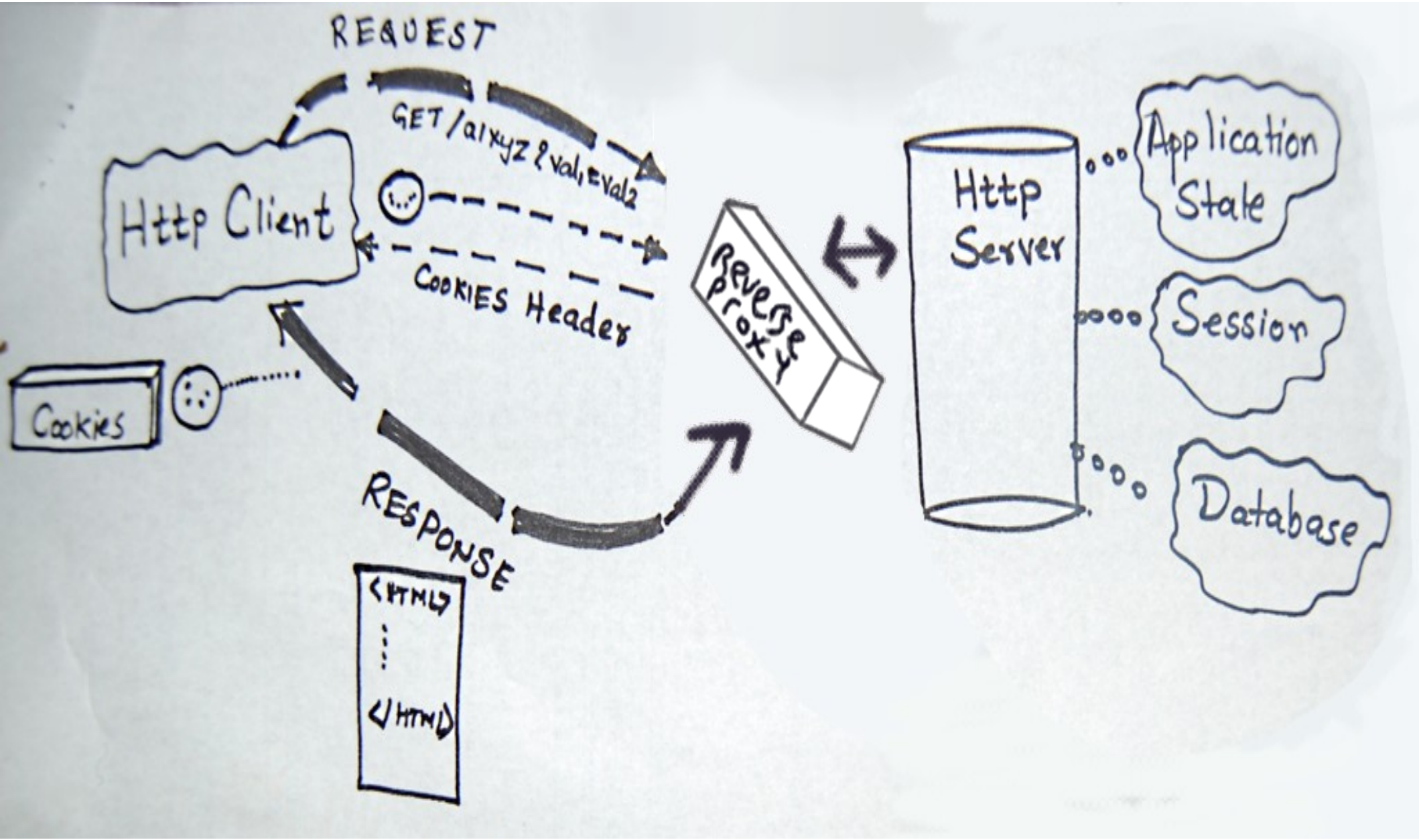
Value:

(function(){var aa="_gat",ba="_gaq",r=true,v=false,w=undefined,ca=document,da="4.7.2",y="length",z="cookie",A="location",ea="_gaUserPrefs",fa="ioo",B="&",C="=",D="__utma=",F="__utmb=",G="__utmc=",ga="__utmk=",H="__utmv=",K="__utmz=",L="__utmx=",ha="GASO=";var M=function(i){return w==i||"-"==i||""==i},ia=function(i){return i[y]>0&&"\n\r\t".indexOf(i)>-1},O=function(i,f,m){var u="-",l;if(!M(i)&&!M(f)&&!M(m)){l=i.indexOf(f);if(l>-1){m=i.indexOf(m,l);if(m<0)m=i[y];u=N(i,l+f.indexOf(C)+1,m)}}return u},ka=function(i){var f=v,m=0,u,l;if(!M(i)){f=r;for(u=0;u<i[y];u++){l=i.charAt(u);m+="."==l?1:0;f=f&&m<=1&&(0==u&&"-"==l||".0123456789".indexOf(l)>-1)}}return f},P=function(i,f){var m=encodeURIComponent;return m instanceof Function?f?encodeURI(i):m(i):escape(i)},

Q=function(i,f){var m=decodeURIComponent,u;i=i.split("+").join(" ");if(m instanceof Function)try{u=f?decodeURI(i):m(i)}catch(l){u=unescape(i)}else u=unescape(i);return u},R=function(i,f){return i.indexOf(f)>-1},S=function(i,f){i[i[y]]=f},U=function(i){return i.toLowerCase()},V=function(i,f){return i.split(f)},la=function(i,f){return i.indexOf(f)},N=function(i,f,m){m=w==m?i[y]:m;return i.substring(f,m)},ma=function(i,f){return i.join(f)},na=function(i){var f=1,m=0,u;if(!M(i)){f=0;for(u=i[y]-1;u>=0;u--){m=

....

Reverse Proxy Placement



Reverse Proxies Are

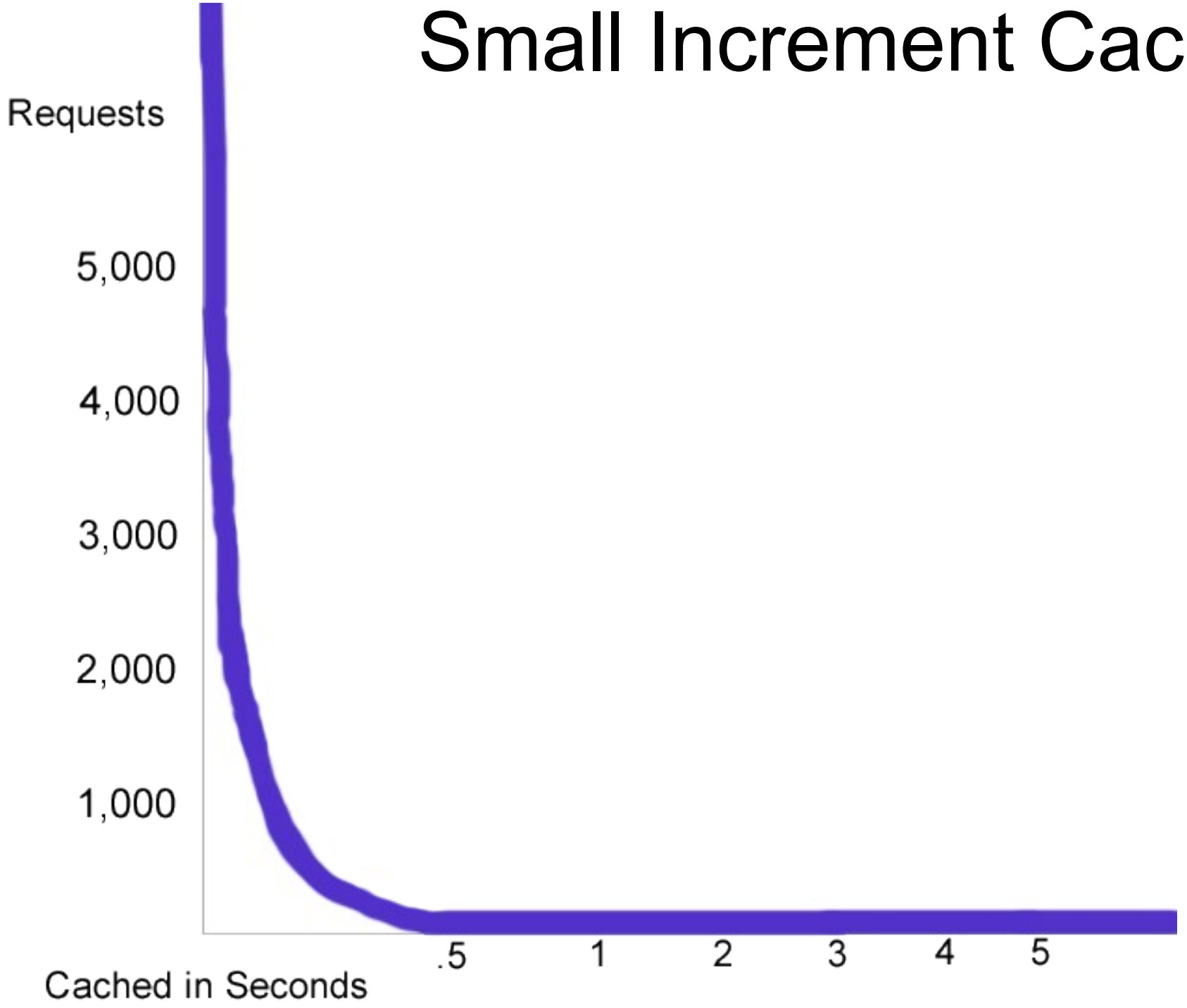
- Very Very Fast

  (5,000 – 15,000 requests per second)
- Great for Static Content
- Pattern Matching
- Highly Reliable
- Cheap (mostly)
- Hates Query Strings (configurable)
- Caches whole response
- Used EVERYWHERE

But, But, That's not Real Time!!!

Users need to see immediately when I ~~deface~~ update the Osama Bin Laden page!

Small Increment Caching



Back to Wikipedia

- 60+ Squid Servers
- 1 000 HTTP requests/s per server, up to 2 500 under stress
- ~ 100 - 250 Mbit/s per server
- ~ 14 000 - 32 000 open connections per server
- Up to 40 GB of disk caches per Squid server
- Up to 4 disks per server (1U rack servers)
- 8 GB of memory, half of that used by Squid
- Hit rates: 85% for Text, 98% for Media, since the use of CARP (Cache Array Routing Protocol)


Free!

Cache Hit:

- When something is requested and already in the cache
- This is an efficiency

Cache Miss:

- When something is requested that is not already in the cache
- This may or may not be an inefficiency

A Cache holds on to something in exchange for memory or disk space. Guess wrong and you waste a scarce resource

But wait,

there's more...

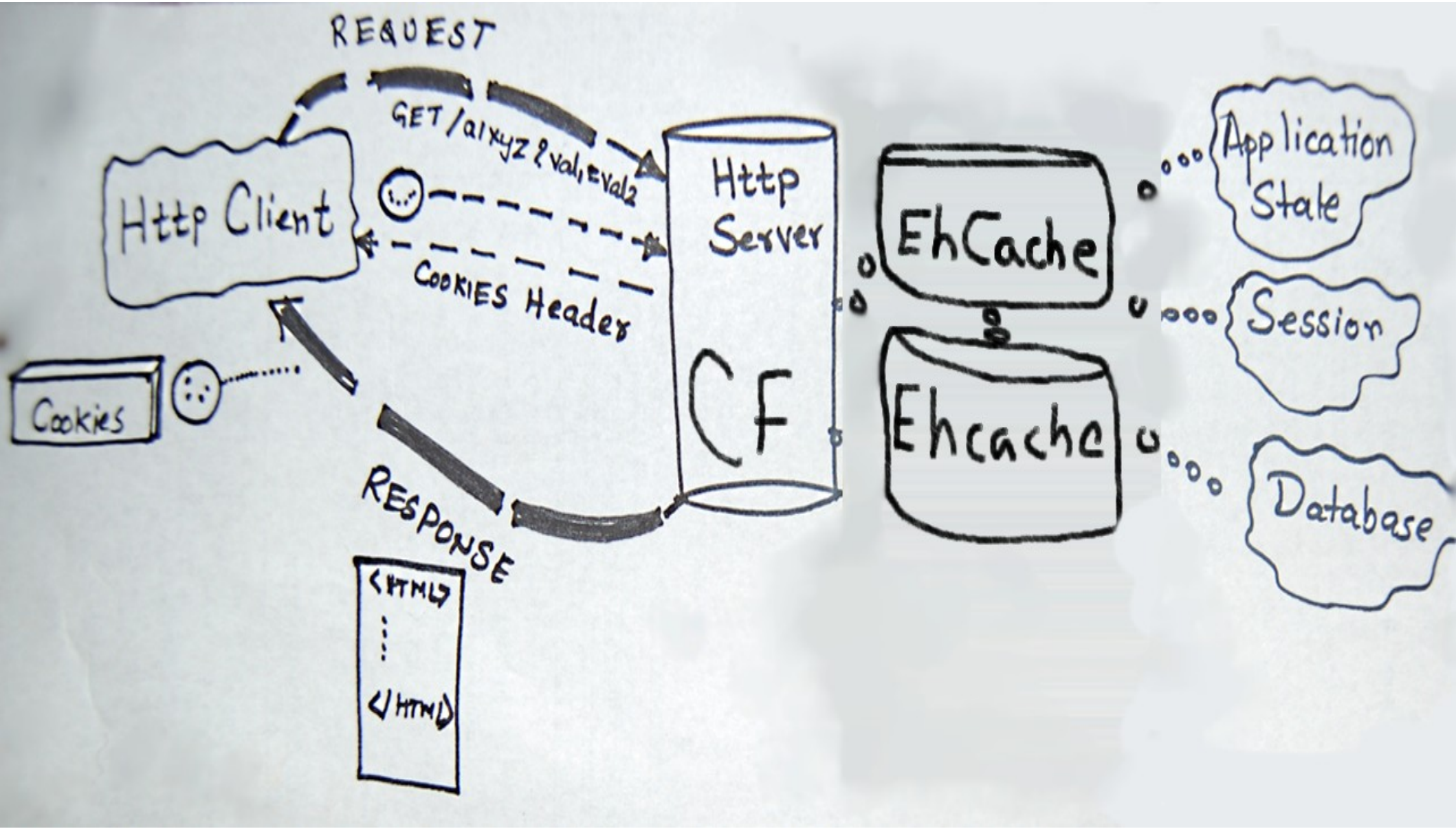
In Memory Cache

- In memory caches are key/value stores that can run in the same memory space as the application or in a different memory space.

  (application scope, session scope, ehCache, memcached)
- Some in memory caches can distribute a large hash across multiple machines

  (ehCache, memcached)

In Memory Cache Placement



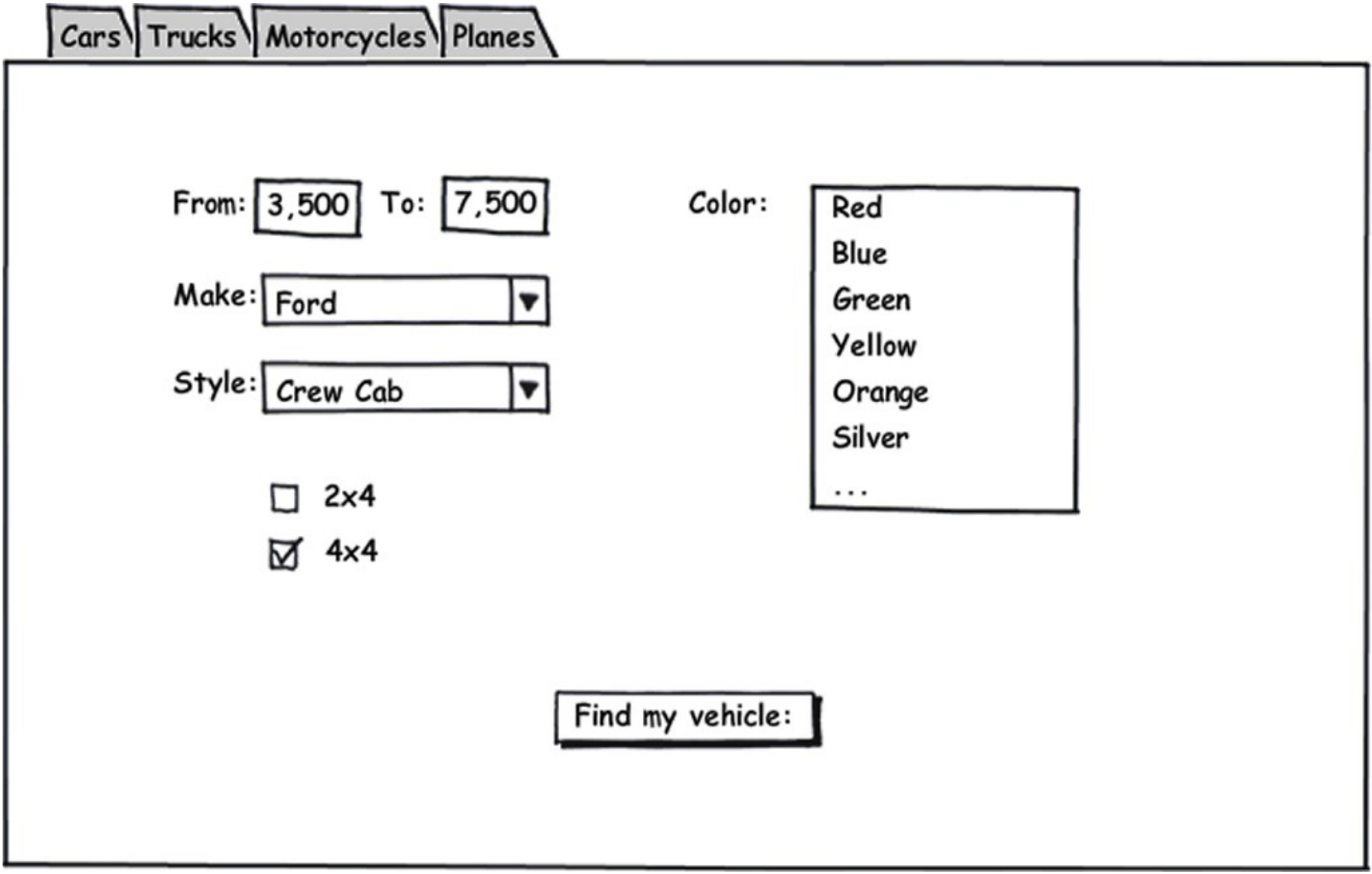
In Memory Cache Code

```
<cffunction name="getListing" output="false">
    <cfargument name="make" type="string" required="true" />
    <cfset var queryKey = "ListingQuery_make#arguments.make#" />
    <cfset var listingQuery = memcached.get( queryKey ) />

    <cfif NOT isQuery( listingQuery )>
        <cfset listingQuery = listingDAO.load( make:arguments.make ) />
        <cfset memcached.set( queryKey, listingQuery ) />
    </cfif>

    <cfreturn listingQuery />
</cffunction>
```

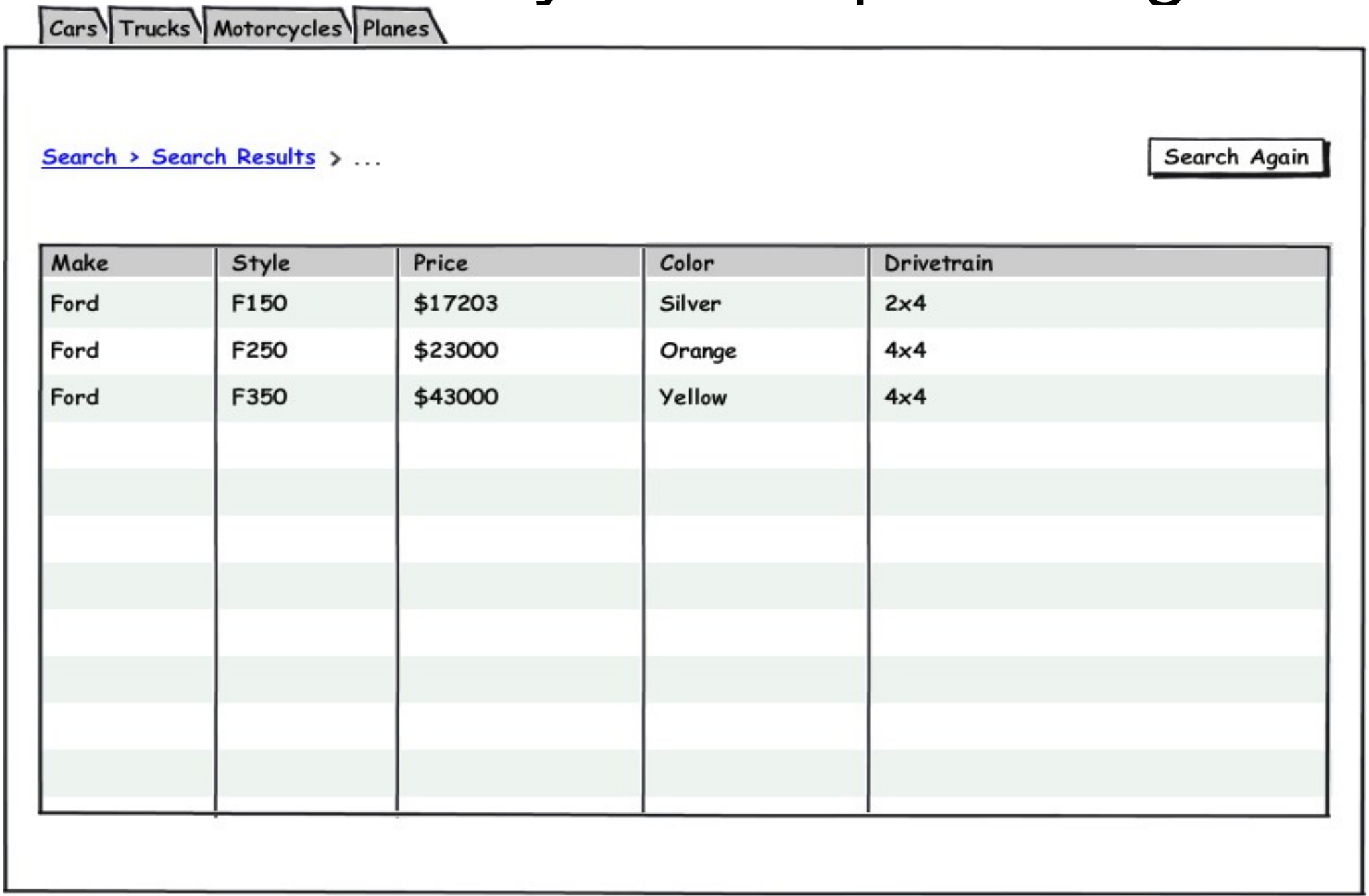
How do we decide what to cache?

Case Study: Vehicle Finder

Case Study


Cars
Trucks
Motorcycles
Planes
From:
3,500
To:
7,500
Color:
Red
Blue
Green
Yellow
Orange
Silver
...
Make:
Ford
Style:
Crew Cab
2x4
4x4
Find my vehicle:

How would you set up caching?



```
<cffunction name="getListing" output="false">
    <cfargument name="make" type="string" required="true" />
    <cfargument name="from" type="string" required="true" />
    <cfargument name="to" type="string" required="true" />
    <cfargument name="style" type="string" required="true" />
    <cfargument name="powertrain" type="string" required="true" />
    <cfargument name="color" type="string" required="true" />
    <cfset var queryKey =
"ListingQuery_make#arguments.make#_from#arguments.from#_to#arguments.to#_s
tyle=#arguments.style#_powertrain#arguments.powertrain#_color#arguments.co
lor#" />
    <cfset var listingQuery = memcached.get( queryKey ) />

    <cfif NOT isQuery( listingQuery )>
        <cfset listingQuery = listingDAO.load( make:arguments.make ) />
        <cfset memcached.set( queryKey, listingQuery ) />
    </cfif>

    <cfreturn listingQuery />
</cffunction>
```

Permutations

20 makes * 6 colors * 3 drive-trains * 4 styles

* 100,000 Low Price possible values

* 100,000 High Price possible values

20*6*3*4*100,000*100,000

= 14,400,000,000,000 @ 1kb each

= 13.4 Petabytes

= Crap.

Yes it can be hard

There are only two hard things in Computer Science: cache invalidation and naming things.

--Phil Karlton

Case Study: Weather Station

Page Generation Process

- User Requests a Page.
- The Page contains a weather alert.
- If the Page exists in memcached, serve the Page to the user. If not, generate the Page, put it in memcached and serve it to the user.
- When a new weather alert arrives, remove all Pages from memcached.
- Under load, the server becomes unresponsive and restarts. Why?

Key considerations when picking the right caching option

Requirement
Easy of integration
Cache API feature set
What uses (templates, ORM, own data, …) – Automatic/Manual
Shared data between multiple machines
Data capacity
Performance
High Availability (for shared data)
Coherency/Drift-free consistency (between machines)
XA Transactions

Key considerations when picking the right caching option

	Local Cache (Ehcache)	Memcached	Enterprise Ehcache
Easy of integration	Out-of-the-box	Simple code changes	Out-of-the-box, Snap-in
Cache API feature set	Rich: eviction, listeners, decorators, …	Simple	Rich: eviction, listeners, decorators, …
What uses (templates, ORM, other data, …) - Automatic/Manual	All	Just 'other data' (not templates/ORM) - manual)	All
Shared data between multiple machines	No*	Yes	Yes
Data capacity	Limited to JVM Heap (~1GB)	Unlimited (up to 1TB?)	Unlimited (up to 1TB)
Performance	Memory speed (~1µsec)	Network Speed (~10ms)	Memory speed (~1µsec) for hotset, network speed (~10ms) for rest
High Availability	No	No	Yes
Coherency/Drift-free consistency	No	No	Yes (configurable option)
XA Transactions	Yes (configurable option)	No	Yes (configurable option)

History of Ehcache and Terracotta

- Both were founded in 2003; both available as Open Source
- Ehcache Hibernate Provider shipped in-kit 2003

  EH = Easy Hibernate
- Terracotta focused on hard problem of high-performance scalability for enterprise applications
- Ehcache evolved in to leading* Java cache – used in the majority of popular frameworks & Java products (eg Spring, Hibernate, Grails, Liferay, Alfresco, Atlassian, ColdFusion, …)

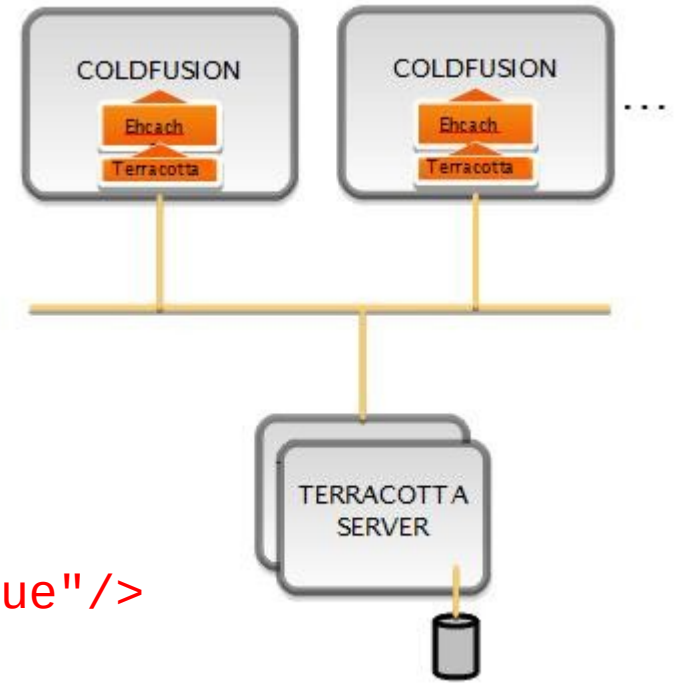
  *70% of Java customers use it according to 2009 Sun Survey
- Terracotta awarded Duke's Choice Award 2009

- Ehcache acquired by Terracotta 2009 & integrated with Terracotta Server to create Enterprise Ehcache
- Ehcache 2.0 March 2010 – adds broad set of features
- Enterprise Ehcache - Forrester Wave "Leader" in Elastic Caching May 2010

Enterprise Ehcache in ColdFusion - 3 easy steps

- Add the ehcache-terracotta jar
- Edit your ehcache.xml

```
<ehcache>
  <terracottaConfig url="someserver:9510"/>
  <defaultCache
     maxElementsInMemory="10000"
     timeToLiveSeconds="120">
  </defaultCache>
  <terracotta clustered="true" coherent=”true"/>
</ehcache>
```



- Start the Terracotta server

```
bin/start-tc-server.sh
```

Performance Comparison

1MM objects	Read-only (100% offload)	Read-write 90% read
Enterprise Ehcache	2 ms 154,587 tps	7 ms 54,197 tps
Memcached	10ms 24,673 tps	10 ms 17,349 tps
Database (MySQL)	106 ms 1,122 tps	95 ms 1,259 tps

- Benchmark's done with standard Java Spring Petstore App
- Configuration used
  - 8 Application nodes (L1s) (32Bit 1.7GB heap)
  - 2 Terracotta/Memcached server pairs (L2s) (32Bit 1.7GB heap)
  - All machines: 4 core 2+GHz machines running RHEL5

44

Helpful Links

In-Memory Cache

http://www.ehcache.org

http://terracotta.org/coldfusion

http://memcached.org

Reverse Proxy Cache

http://varnish-cache.org

http://www.squid-cache.org

Tutorials on Caching

Varnish: http://bit.ly/c1puD6

Squid Video: http://bit.ly/9iZu1Z

Aaron West: http://bit.ly/a4sYcr

Rob Brooks-Bilson: http://bit.ly/txser

Terracotta Webinar: http://www.terracotta.org/webcasts

This presentation: http://tinyurl.com/cacheme

Scalability Blogs

http://cfwhisperer.com

http://highscalability.com

Thanks

Dan Wilson

twitter.com/DanWilson

nodans.com

datacurl.com

challengewave.com

model-glue.com

Mike Brunt

twitter.com/cfwhisperer

cfwhisperer.com

go2ria.com