The Top 5 Things **You** are doing today to hinder scalability

Dan Wilson

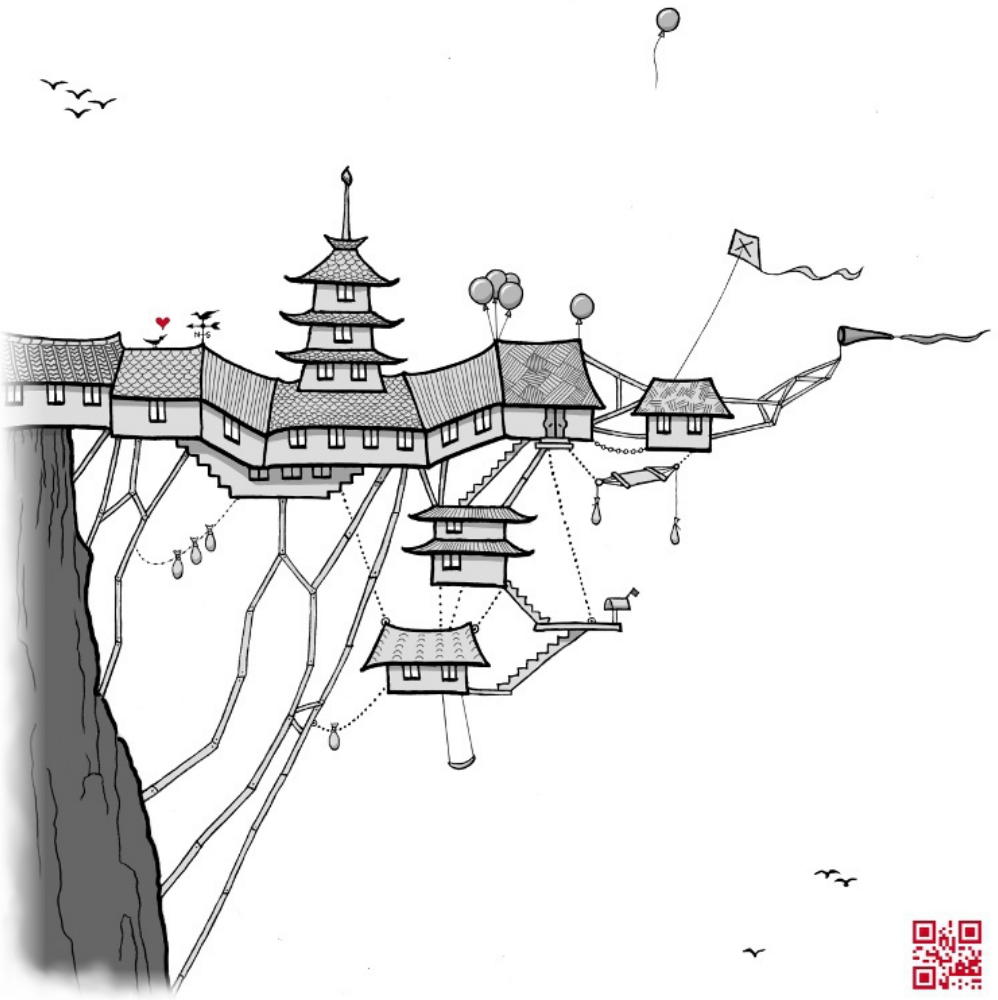
Who is Dan Wilson

- Runs ChallengeWave.com, Datacurl LLC and NCDevCon
- Technical Owner of Scholastic.com, Lead Architect of Forum Group
- Owner of Model-Glue
- Dad of 2, Husband to 1
- Loves all things technical and a good debate

Our Top 5 List

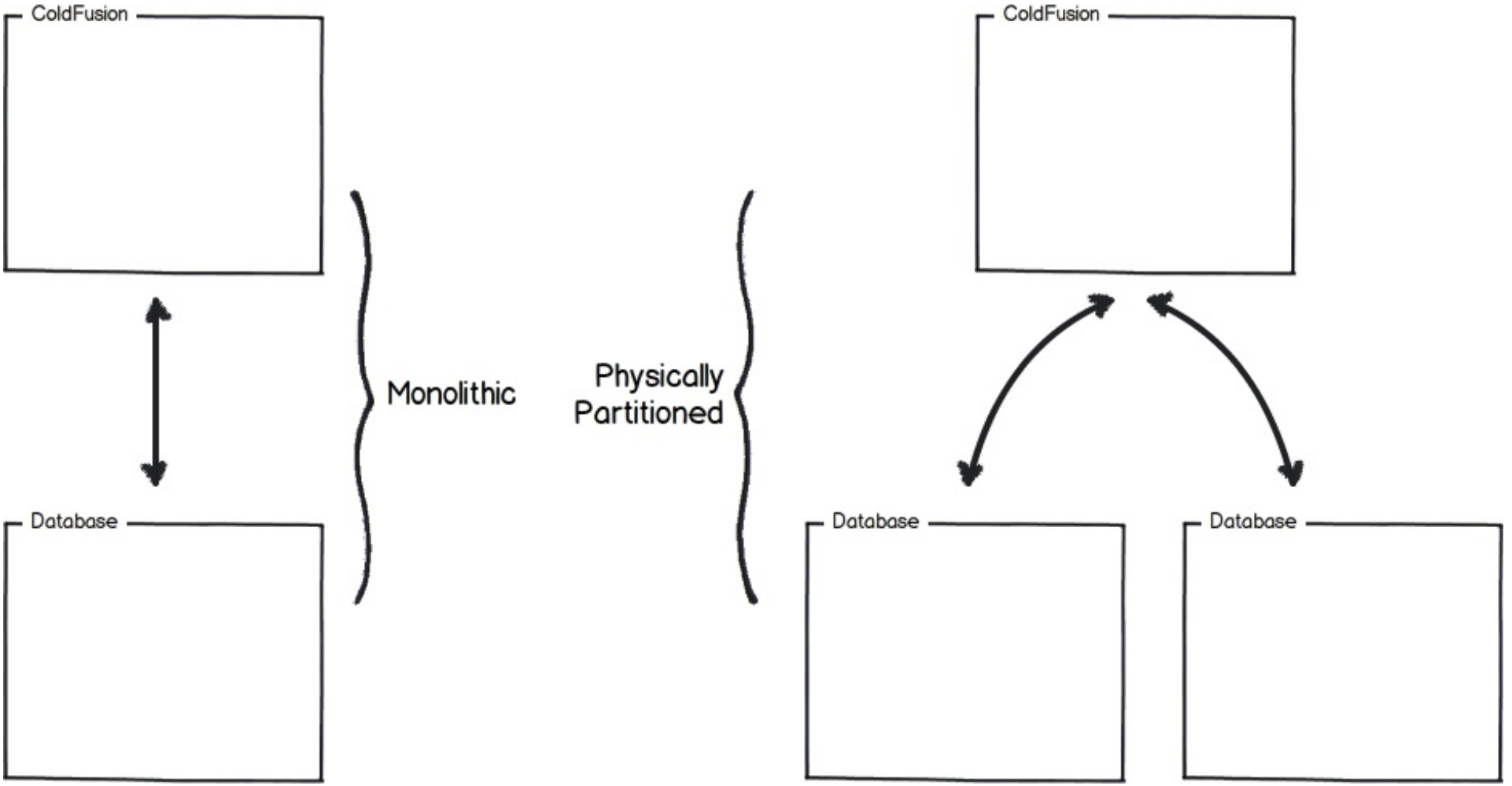
1. Monolithic Design
2. Mis-identification/Misuse of shared resources
3. Over-reliance on Shared Scopes
4. Stupid Developer Tricks
5. Failure to design for cacheability

Monolithic Design

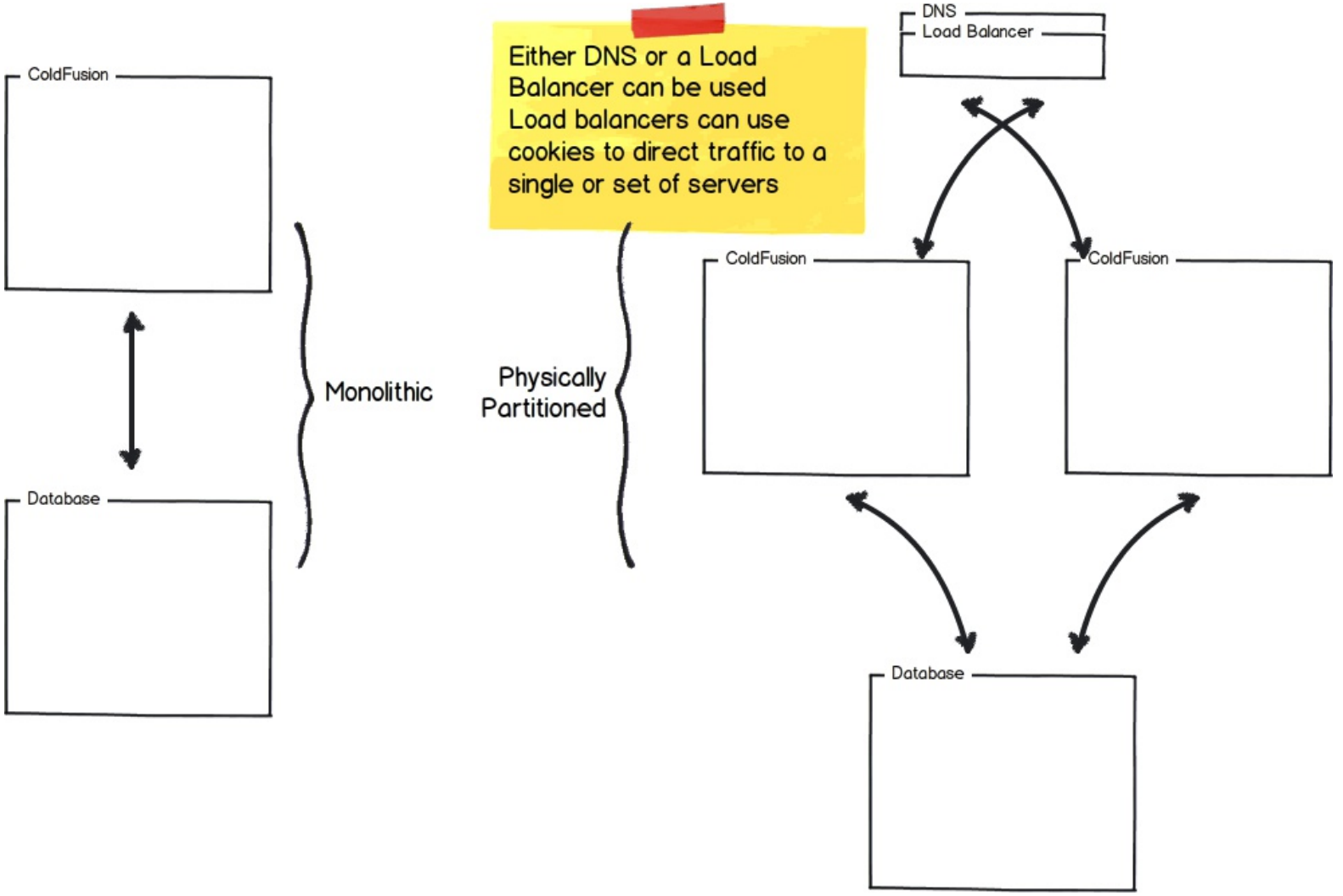


Credit: http://thecodelesscode.com/case/33

Solution: Partition Databases



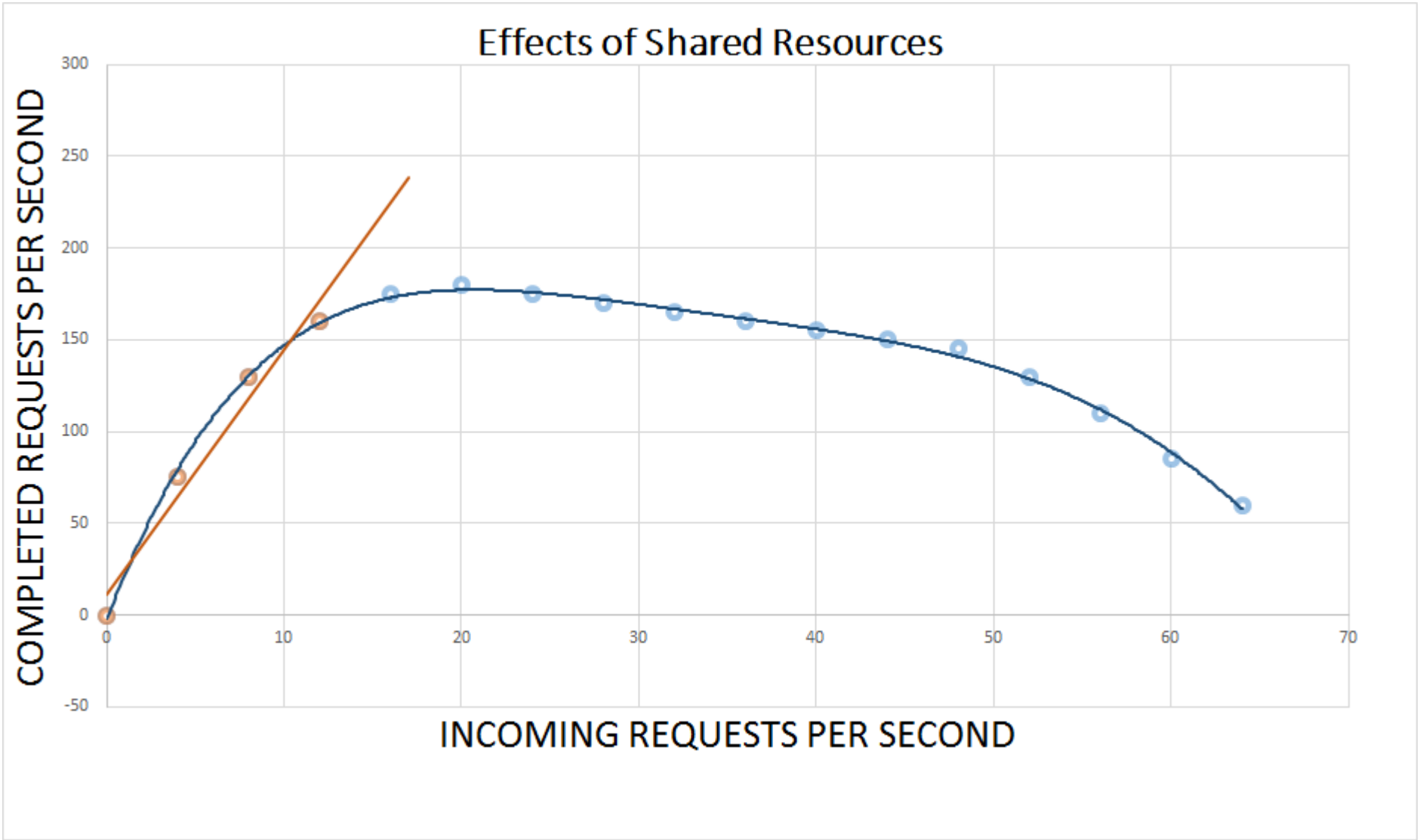
Solution: Decouple Functionality




FOOD TRUCK FESTIVALS
OF NEW ENGLAND
xfinity

Shared Resources

Scalability Curve



Fun With Math

$$t = \frac{1}{a}\left(v_f - v_i\right) \quad \text{substitute into} \quad x_f = x_i + v_i t + \frac{1}{2}at^2$$

$$x_f = x_i + \frac{v_i}{a}\left(v_f - v_i\right) + \frac{1}{2}a\left[\frac{1}{a}\left(v_f - v_i\right)\right]^2$$

$$ax_f = ax_i + v_i\left(v_f - v_i\right) + \frac{1}{2}\left[\left(v_f - v_i\right)\right]^2$$

$$2ax_f - 2ax_i = 2v_i\left(v_f - v_i\right) + \left(v_f - v_i\right)^2$$

$$2a(x_f - x_i) = 2v_i v_f - 2{v_i}^2 + {v_f}^2 - 2v_i v_f + {v_i}^2$$

$$2a(\Delta x) = {v_f}^2 - {v_i}^2$$

Cost of WebServer File

Web Server Thread

+ Network to File System

+ Disk Wait Time

+ Disk Seek Time

+ Transfer time to Webserver

+ Transfer time to complete HTTP Request

Total Time

Cost of 1 ColdFusion File

Web Server Cost

+ Network Cost to CF Server

+ CF Parsing or Classloading Time

+ Thread Wait time

+ Thread CPU time

+ Transfer time for CF buffer to Webserver

Total Time

Cost of ColdFusion File With Query

Web Server Cost

+ ColdFusion File Cost

+ Number of Queries (Network time to DB + Query Execution Plan + Query Execution + Disk Wait + Disk Seek)

+ Network time to transport result set to CF

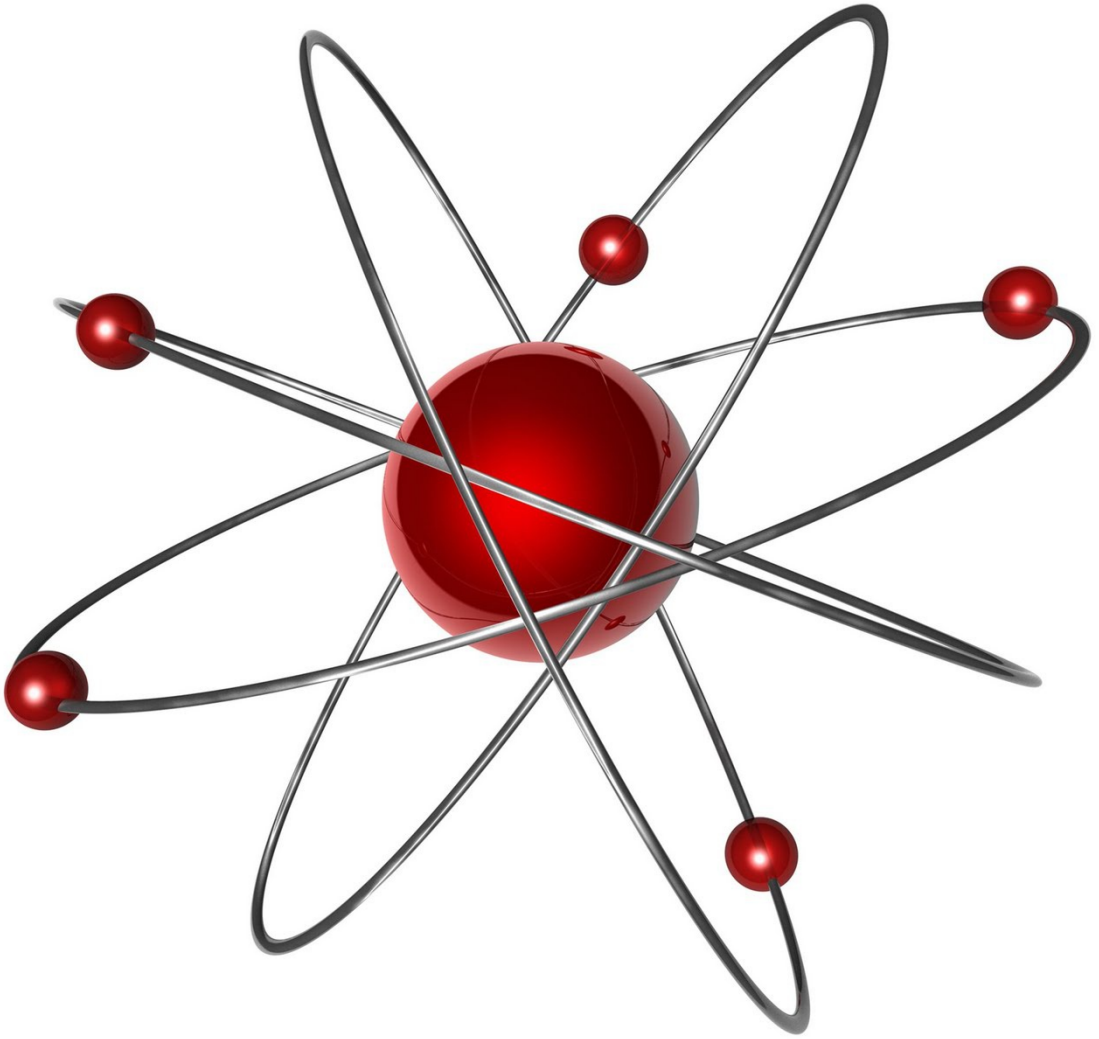
+ CF Parsing Time

Total Time

Types of Shared Resources

- Memory, CPU, Disk Space
- Disk Transfer (Databases!!)
- JVM Space
- ColdFusion template cache
- ColdFusion Threads
- Network Traffic
- Database

DANGER! DANGER!

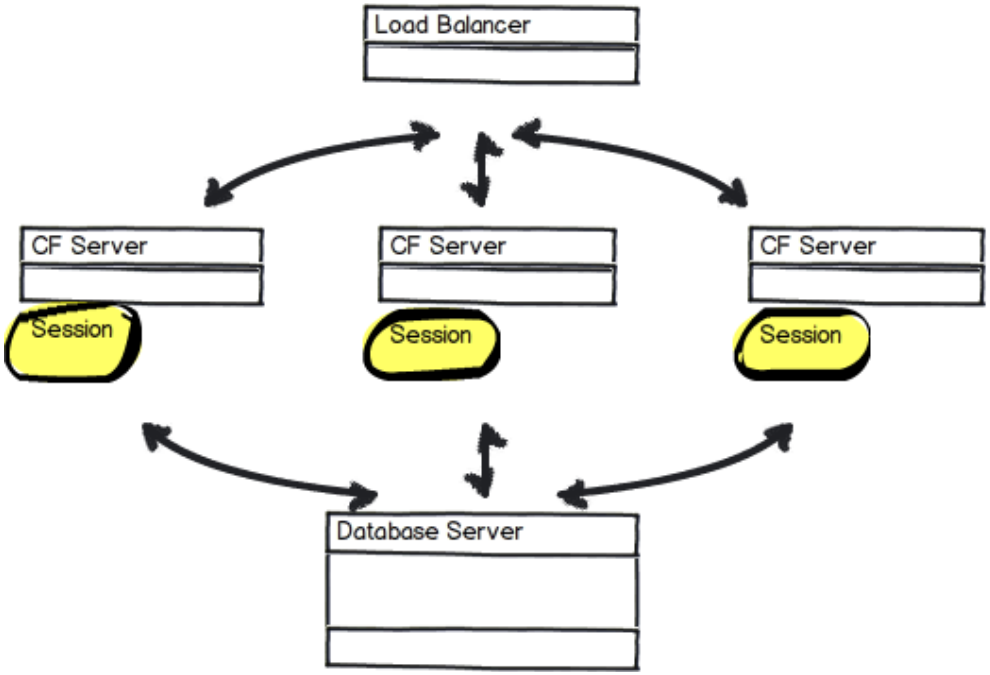


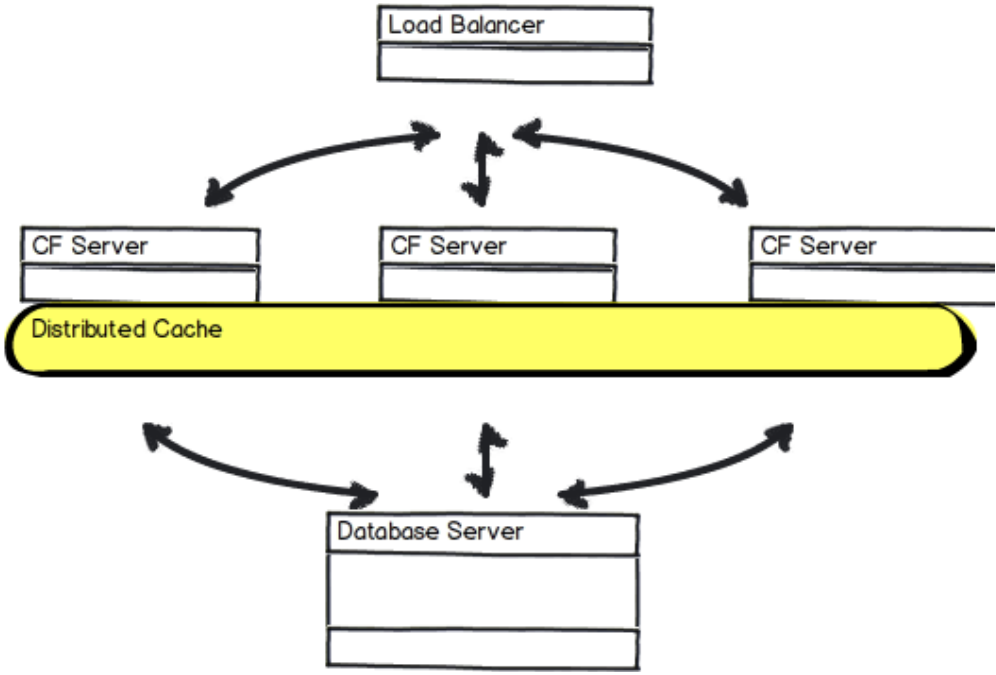
Over-reliance on Shared Scopes

- Session Scope use is the largest culprit
- Sometimes you have to deoptimize a few operations in order to provide a scalable solution (like stop caching a state query in application)
- If you want to use a shared scope and want to keep an eye on scalability later, use a lookup factory.

```
<cfcomponent>
	<cfset variables.instance = structNew() />

	<cffunction name="init" returntype="CurrentUserLoader" output="false"
access="public">
		<cfargument name="Transfer" type="any" required="true"/>
		<cfset variables.transfer = arguments.transfer />
		<cfreturn this />
	</cffunction>

	<cffunction name="destroy" output="false" access="public" returntype="void" >
			<cfset cookie.userID = "" />
			<cfset structDelete( cookie, "userID") />
	</cffunction>

	<cffunction name="load" output="false" access="public" returntype="any" hint="">
		<cfset var loggedInUserID = 0 />
		<cfif structkeyExists(cookie, "userID")>
			<cfset loggedInUserID = cookie.userID />
		</cfif>
		<cfreturn variables.transfer.get("User", val( loggedInUserID ) ) />
	</cffunction>

	<cffunction name="set" output="false" access="public" returntype="void" hint="">
		<cfargument name="user" type="any" required="true"/>
		<cfset cookie.UserID = arguments.User.getUserID()  />
		<cfset cookie.Email = arguments.User.getEmail() />
	</cffunction>
</cfcomponent>
```

Avoiding Server Shared Scopes





Stupid Developer Tricks

- Looped Queries
- Lazy-Man Pagination
- ScheduledTask-o-rama
- Database Abuse

Problem: Query in Loop

```
<cfloop query="OldLastLogins">
    <cfquery name="UpdateUsers" datasource="#datasource#">
          UPDATE Users
          SET Active = 0
          WHERE UserID = #OldLastLogins.UserID#
    </cfquery>
</cfloop>
```

Solution?: SQL IN Statement

```
<cfquery name="UpdateUsers" datasource="#datasource#">
   UPDATE Users
   SET Active = 0
   WHERE UserID IN #ValueList(OldLastLogins, UserID)#
</cfquery>
```

Solution: Joins

```
<cfquery name="UpdateUsers" datasource="#datasource#">
    UPDATE users
       INNER JOIN lastlogin ON users.userID = lastlogin.userID

       SET Active = 0
    WHERE lastlogin < now()
</cfquery>
```

Solution: Joins

```
<cfquery name="UpdateUsers" datasource="#datasource#">
    UPDATE users
        INNER JOIN lastlogin ON users.userID = lastlogin.userID

        SET users.lastLoginDate = lastlogin.createdDate
    WHERE lastlogin < now()
</cfquery>
```

Problem: EntityToQuery

```
<cfscript>
   EntityToQuery(
      EntityLoad("Users",
         { Gender = arguments.genderID}
      )
   );
</cfscript>
```

Example: Get Poll Answers

```
<cfset PollAnswers = entityLoad("PollAnswers",
                {
                   PollID=entPoll[1].getPollID()
                }, "")>


<cfif (isArray(Poll) AND arrayLen(Poll) GT 0)
   AND (isArray(PollOpt)
   AND arrayLen(PollOpt) GT 0)
   AND (isArray(PollAnswers)
   AND arrayLen(PollAnswers) GT 0)>
```

Poll Net Result

JDBC

Recordcount: 87,923

Query Time: 14,290 ms

All that for an ArrayLen() GT 0!

Solution: Query

```
<cfquery name="LoadGenderUsers" datasource="#datasource#">
   SELECT UserID, UserName, Email, GenderID
   FROM Users
   WHERE GenderID = '#arguments.GenderID#'
</cfquery>
```

Problem: Lazy Man Pagination

```
<cfoutput query="GetParks"
        startrow="#StartRow#"
        maxrows="#MaxRows#">
   #ParkName#
</cfoutput>
```

Solution: SQL

MySql, PostgreSQL: LIMIT, OFFSET

SQL Server: TOP, ORDER BY

ORACLE: WHERE rownum <= 10

Problem: No Encapsulation

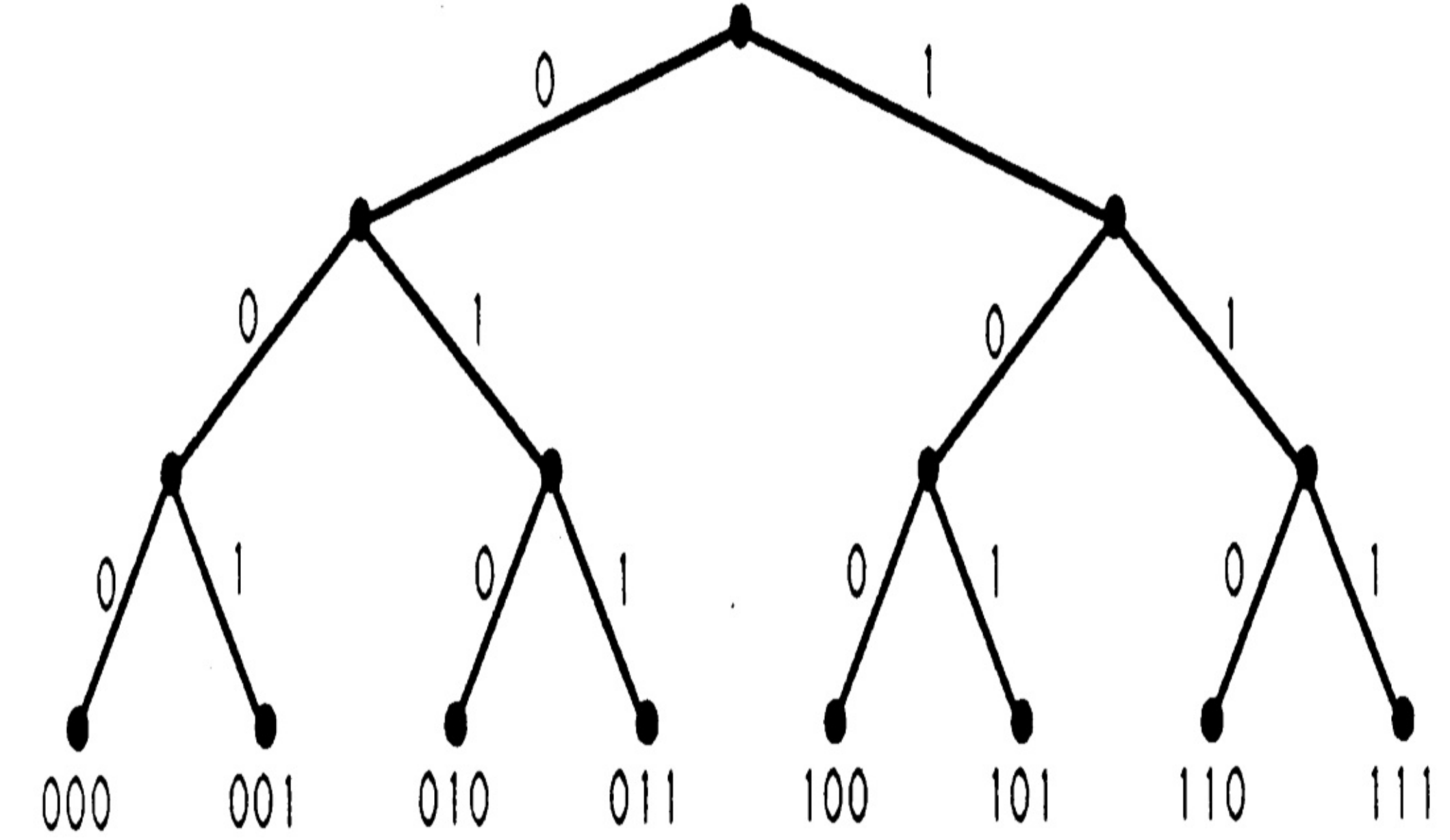
```
<cfif qExists.idexists>
  <cfquery name="qUpdate" datasource="#somedatasource#">
    UPDATE  tblKalendarCategories
    SET CategoryName = '#form.CategoryName#',
      CategoryTextColor = '#form.CategoryTextColor#'
    WHERE CategoryID = #form.CategoryID#' />
  </cfquery>
<cfelse>
  <cfquery name="qCreate" datasource="#somedatasource#">
    INSERT INTO tblKalendarCategories
      ( CategoryName, CategoryBGColor, CategoryTextColor )
    VALUES
     ( '#form.CategoryName#',  '#form.CategoryTextColor#')
  </cfquery>
</cfif>
```

Solution: Encapsulate Data Access

```
DAO.read();
DAO.delete();
DAO.exists();
```

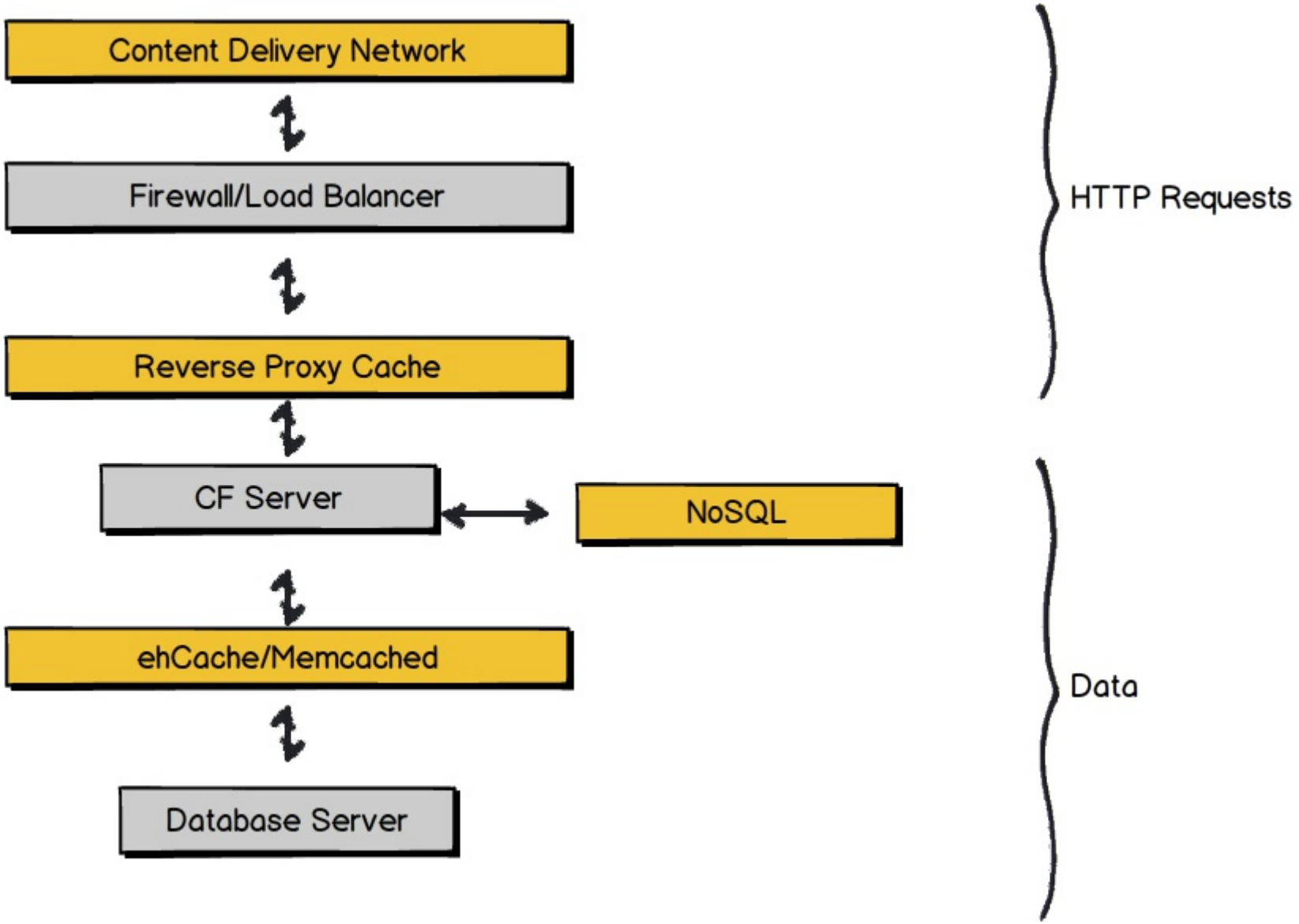
```
DAO.save();
DAO.create();
DAO.update();
```

Bad or No Indexing



Cacheability

Caching Layers



Caching HTTP Responses

Request:

GET /index.html HTTP/1.1

Host: www.example.com

Response:

HTTP/1.1 200 OK

Date: Mon, 23 May 2005 22:38:34 GMT

Server: Apache/1.3.3.7 (Unix) (Red-Hat/Linux)

Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT

ETag: "3f80f-1b6-3e1cb03b"

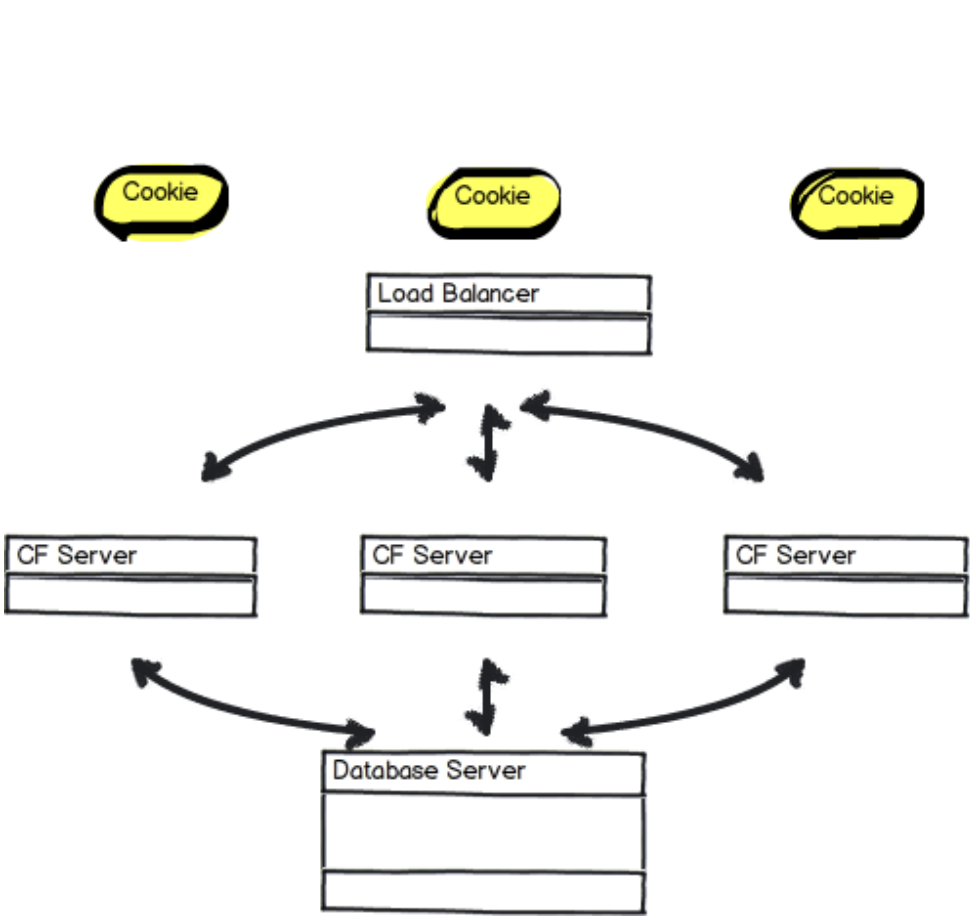
Content-Type: text/html; charset=UTF-8

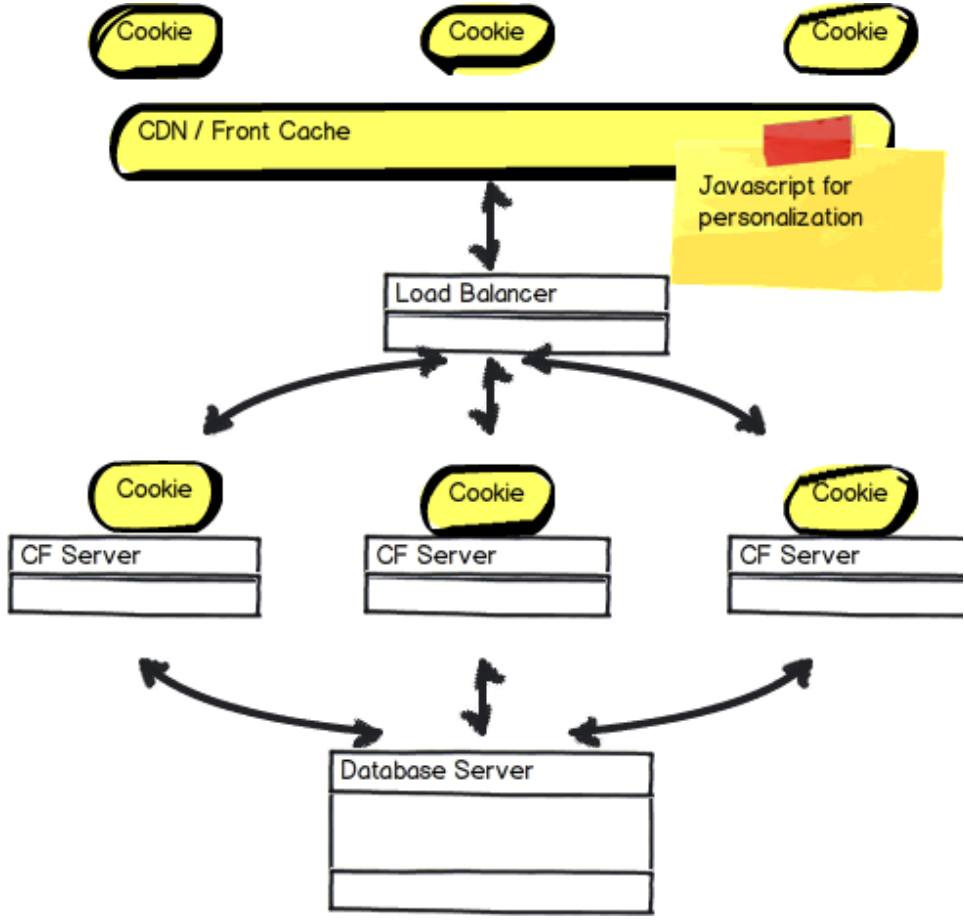
Content-Length: 131

Accept-Ranges: bytes

Connection: close

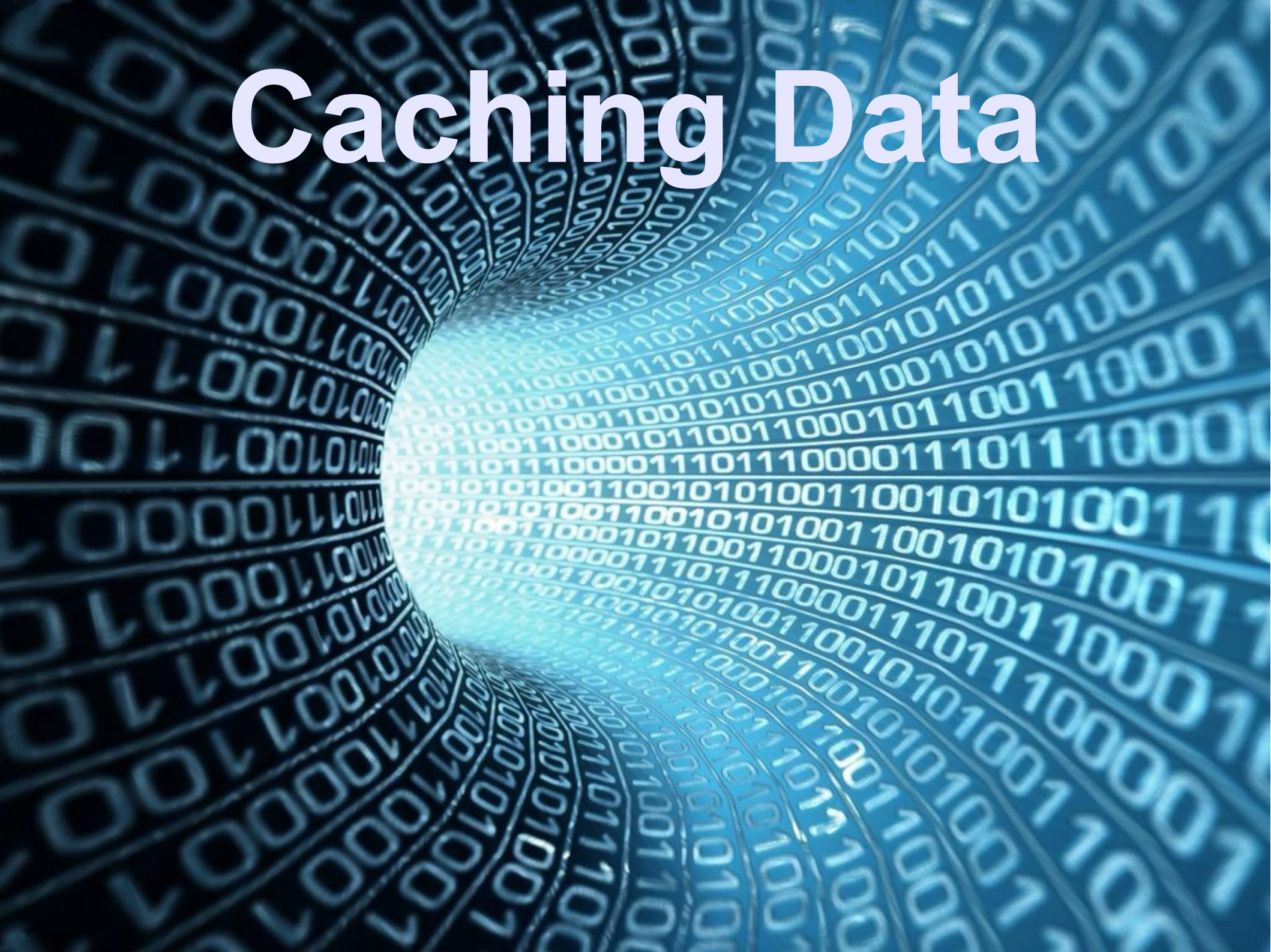
Caching HTTP Responses





Designing for a CDN or Reverse Proxy Cache

- Use HTTP verb “GET” for cacheable content
- Use Last-Modified and Etag headers properly
- Encapsulate code that changes content to centralize cache purge activity
- Personalize with Cookies/Javascript
- Set up shadow DNS to bypass caching
- Use Javascript for interactions
- Design data for high cache hit ratios


Caching Data

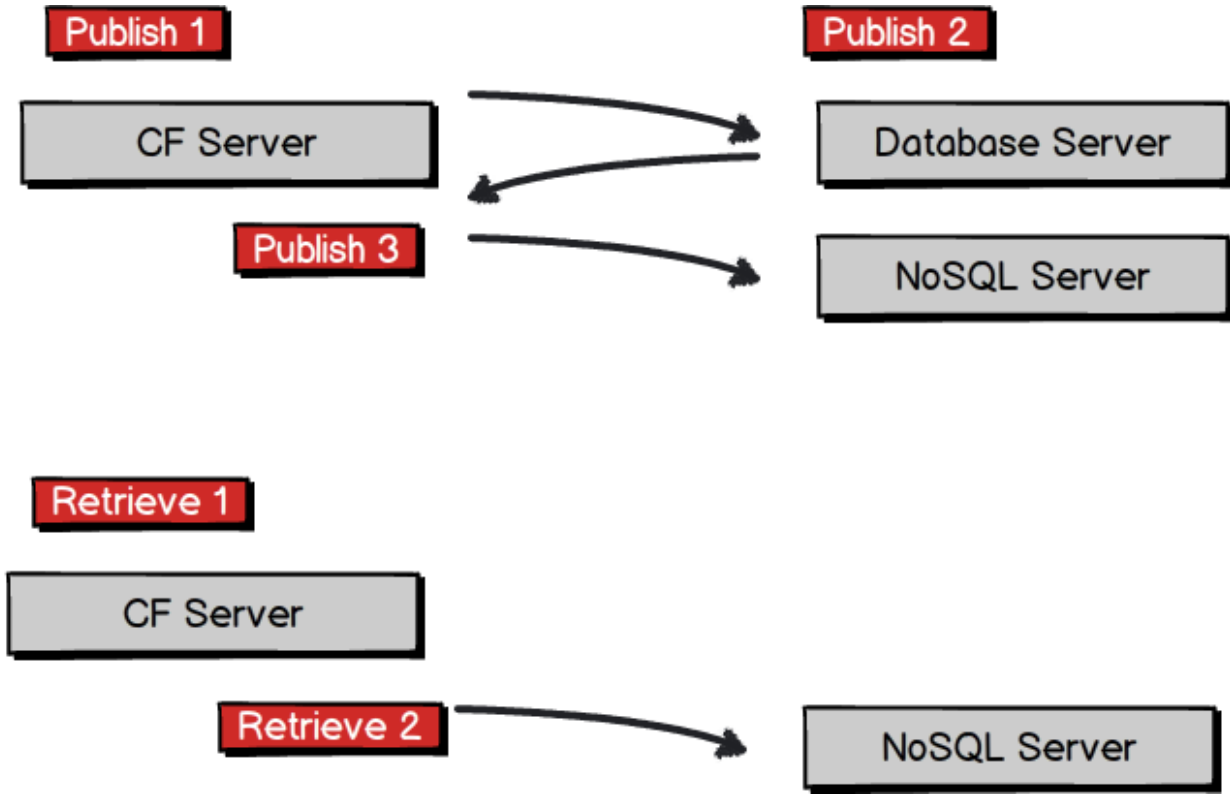
NoSQL as a Cache

```
{
    "_id" : ObjectId("51fff472e09a41ac19dd1876"),
    "NAME" : "Dan",
    "KIDS" : [
        {
            "NAME" : "Nick",
            "AGE" : 2,
            "HAIR" : "blonde"
        },
        {
            "NAME" : "Jack",
            "AGE" : 1,
            "HAIR" : “none”,
        }
    ]
}
```

NoSql Cache Diagram

Keys:
IssueId (120 in last 6 months 1047 overall)-
Start (20
Stop (20)
Lexile (2)
Speed (2)

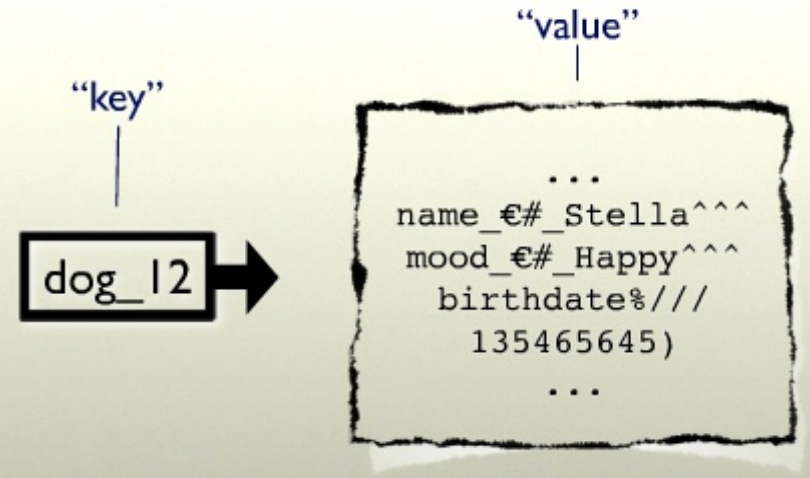
Issues x Start x Stop x Lexile x Speed
= 192000



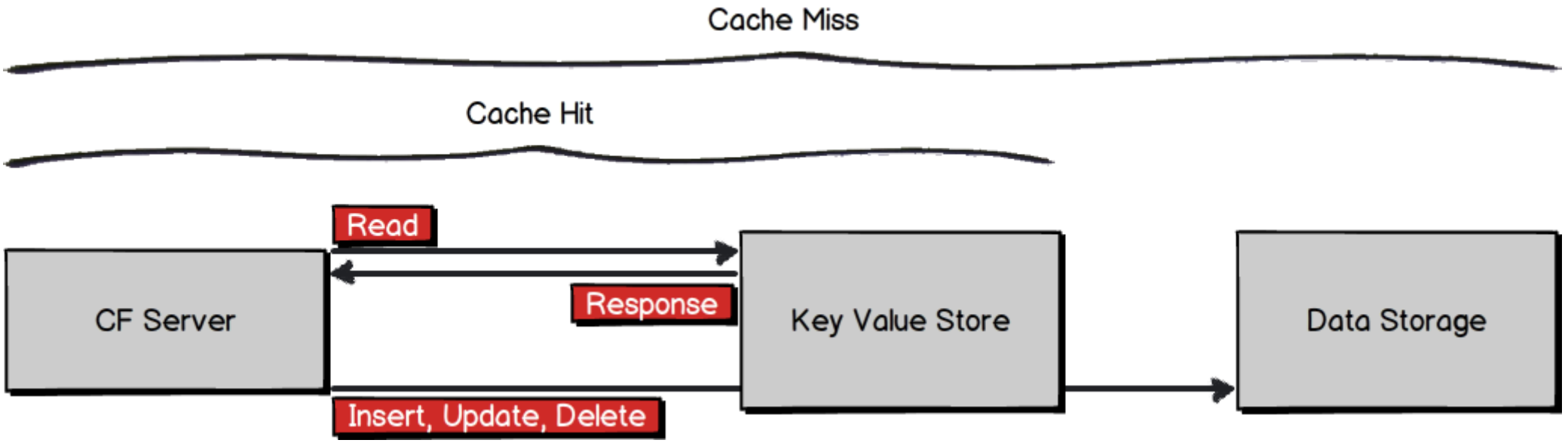
Designing for a NoSQL Cache

- Encapsulate calls to NoSQL engine
- Encapsulate data publish methods
- Design Content Purge algorithms well
- Ensure the NoSQL engine is the **authoritative source** for the data being stored

Key/Value stores



Key Value Store Cache Diagram



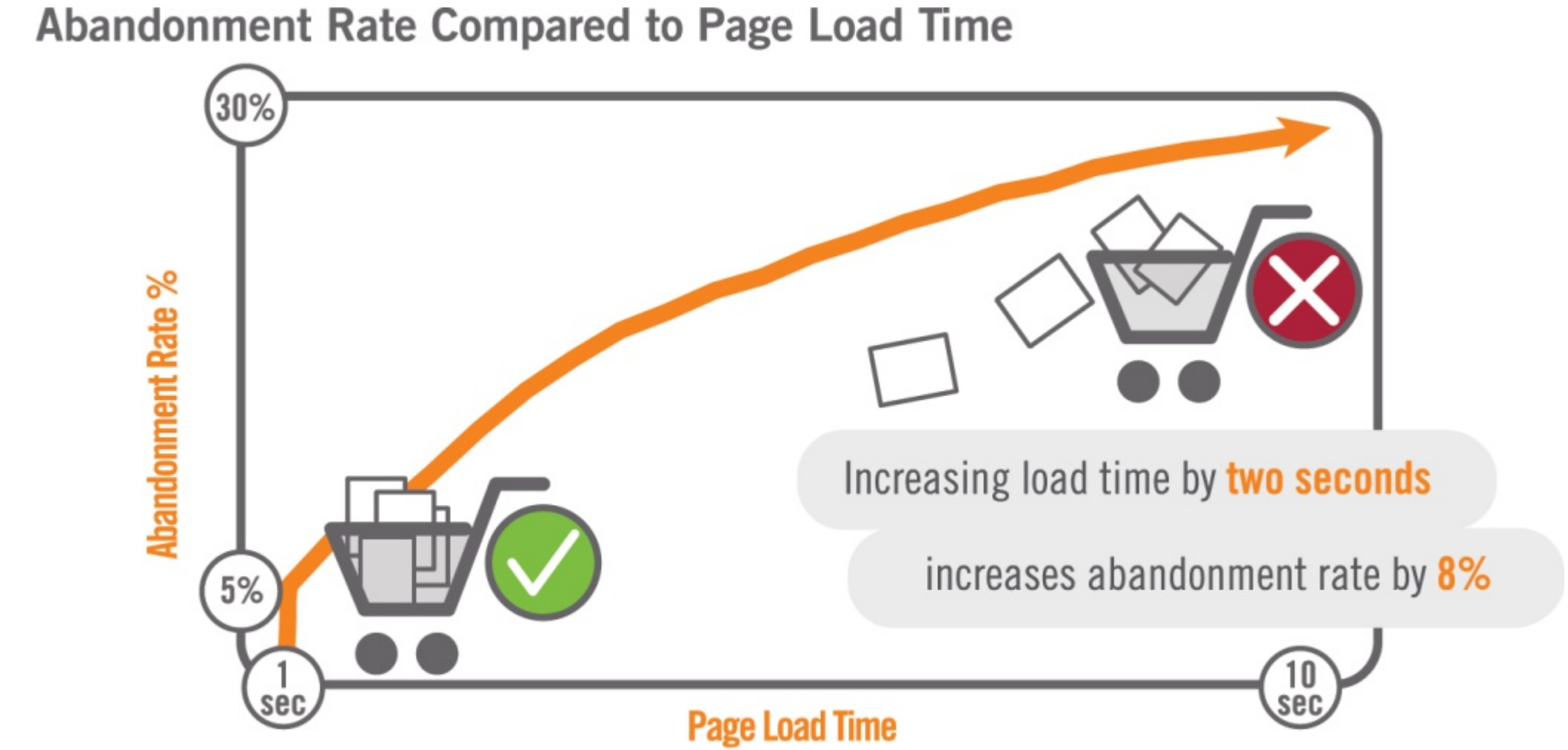
Designing for a Key Value Store Cache

- Encapsulate data access calls so you can purge/reload
- Avoid cffunction output=”true”
- Choose correct cache expiry strategy (time, LRU)
- Easy to integrate with CF ORM
- Use to remove load hotspoting

```
<cfcomponent>
  <cffunction name="loadProduct" returntype="Product" access="public">
    <cfargument name="ProductID" type="numeric" required="true"/>
    <cfset var CacheKey = makeProductCacheKey(arguments.ProductID)/>
    <cfif variables.Cache.exists(CacheKey) IS false>
      <cflock name="#CacheKey#" timeout="10">
        <cfif variables.Cache.exists(CacheKey) IS false>
          <cfset variables.Cache.put(CacheKey,
variables.ProductService.get(arguments.ProductID))/>
        </cfif>
      </cflock>
    </cfif>
    <cfreturn variables.Cache.get(CacheKey)/>
  </cffunction>

  <cffunction name="saveProduct" returntype="Product" access="public">
    <cfargument name="ProductID" type="numeric" required="true"/>
    <cfargument name="Name" type="text" required="true"/>
    <cfset var CacheKey = makeProductCacheKey(arguments.ProductID)/>
    <cfset var Product = variables.ProductService.get(arguments.ProductID)/>
    <cfset Product.setName(arguments.Name)/>
    <cfset Product = variables.ProductService.save(Product)/>
    <cfset variables.Cache.set(CacheKey, Product)/>
    <cfreturn variables.Cache.get(CacheKey)/>
  </cffunction>

  <cffunction name="makeProductCacheKey" returntype="string" access="private">
    <cfargument name="ProductID" type="numeric" required="true"/>
    <cfreturn "Product_#val(arguments.ProductID)#"/>
  </cffunction>
</cfcomponent>
```

Alternate Cache Methods

Time is Money



Source: Gomez Real-User Monitoring
- 250+ customers
- 100,000,000+ page measurements

July 2011

Thanks

Dan Wilson

nodans.com

twitter.com/DanWilson

www.linkedin.com/in/profileofdanwilson

Appendix and helpful links

A Note for MySQL users

[mysqld]

innodb_file_per_table

1. Do a mysqldump of all databases, procedures, triggers etc except the mysql and performance_schema databases
2. Drop all databases except the above 2 databases
3. Stop mysql
4. Delete ibdata1 and ib_log files
5. Start mysql
6. Restore from dump

A Note for MySQL users

[mysqld]

innodb_file_per_table

1. Do a mysqldump of all databases, procedures, triggers etc except the mysql and performance_schema databases
2. Drop all databases except the above 2 databases
3. Stop mysql
4. Delete ibdata1 and ib_log files
5. Start mysql
6. Restore from dump

Content Delivery Network

PROS

- Caches HTTP Response
- Ajax Friendly
- Static Content
- TTL/Manual Purge
- Blazing Fast
- Can be geographically distributed
- No physical infrastructure to manage
- Pay for what you use

CONS

- Pay for what you use
- SSL is more complicated
- Testing is more complicated
- Reliant on external service
- Not a good fit for intranet traffic

Content Delivery Options

www.akamai.com

aws.amazon.com/cloudfront

Monitoring Options

ColdFusion Server Monitor

CFStat

New Relic – SaaS

Fusion Reactor Monitor

Nagios

Reverse Proxy Cache (Front Cache)

traffic server

Reverse Proxy Cache

PROS

- Caches HTTP Response
- Ajax Friendly
- Static Content
- TTL/Manual Purge
- Blazing Fast
- Many Open Source Products
- Low operating expense
- Can be used for intranets

CONS

- Personalization must be client side (no session scope)
- Content is treated as static
- Testing is more complicated
- Must Manage Physical Infrastructure
- Often Geographically Localized

Key Value Store

PROS

- Blazing Fast
- Can cache any in-memory item
- Some caches are sophisticated, some aren't
- Can be distributed
- Simple interfaces
- Integrated in CF

CONS

- Many Cached Items represent an HTTP request
- What is real-time to your application?
- Key management can be hard
- WYCIWYG (what you cache is what you get)

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

http://trafficserver.apache.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