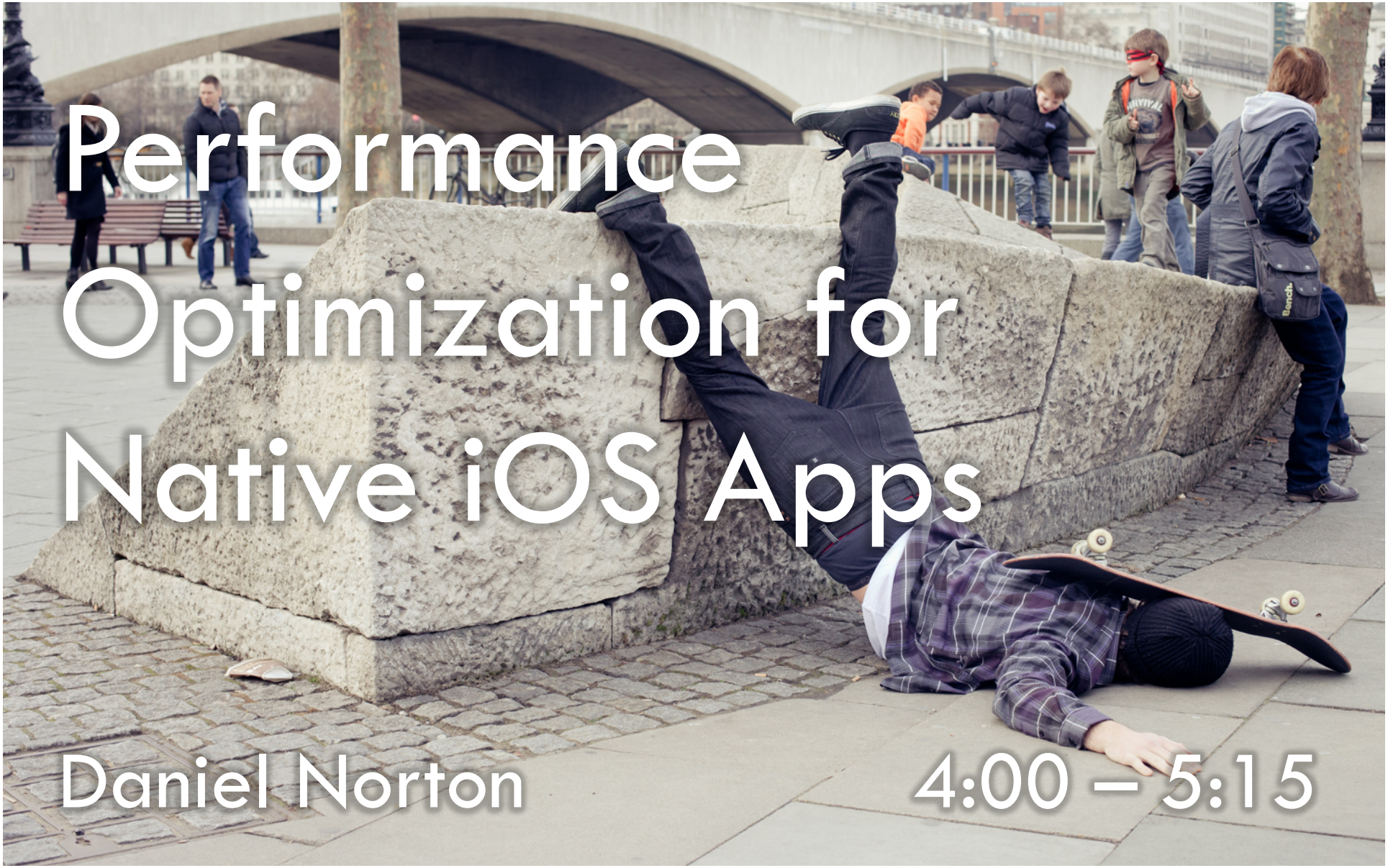

Performance
Optimization for
Native iOS Apps
Daniel Norton
4:00 – 5:15

github: danielnorton
werk: leankit.com
twitter: @daniel_norton

GitHub

github.com/danielnorton/GitHubReader

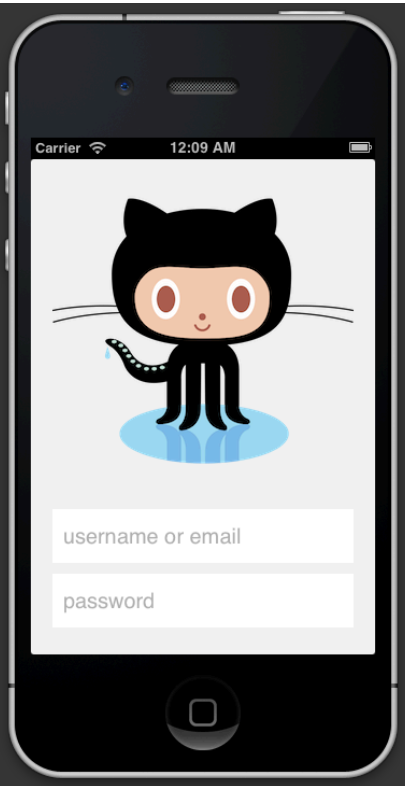
## The Demo App

# Goals for Today

- Detective work
- Iterate to improve a poorly performing app
- Experiment to gain insight
- Use the demo code as a sandbox for additional learning
  - Fork the repo (I'll show it again later)
  - follow the tag suggestions in this slide deck
  - Run your own experiments later

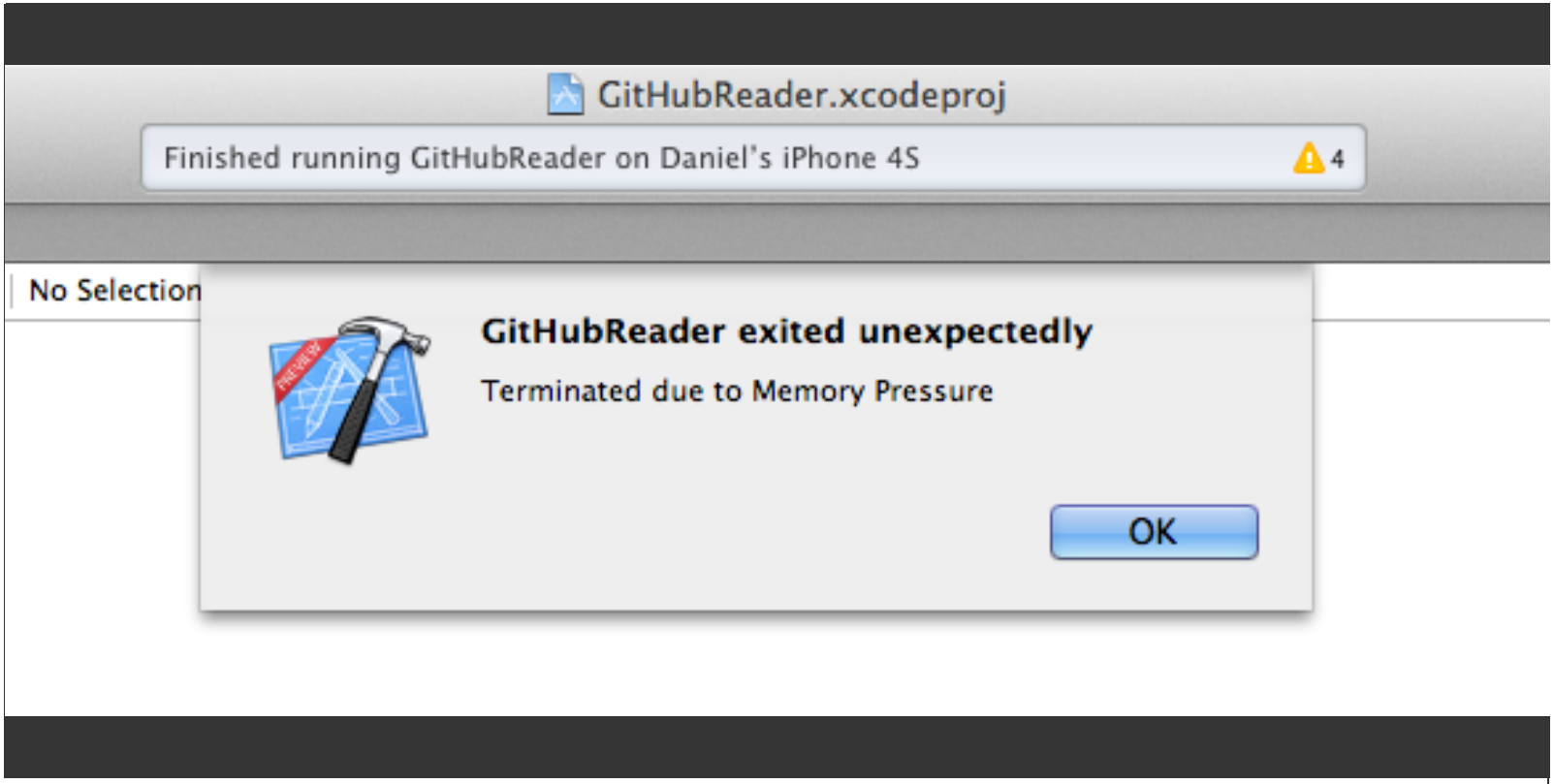
# Tools at Our Disposal

- Xcode
- Instruments
- Multithreading & Networking
- Core Data
- Caching
- Perceived Performance




# git checkout 01_perf_baseline

- Quick tour through the app




# ... a little while later

Well, try it again

# What do we intuitively see?

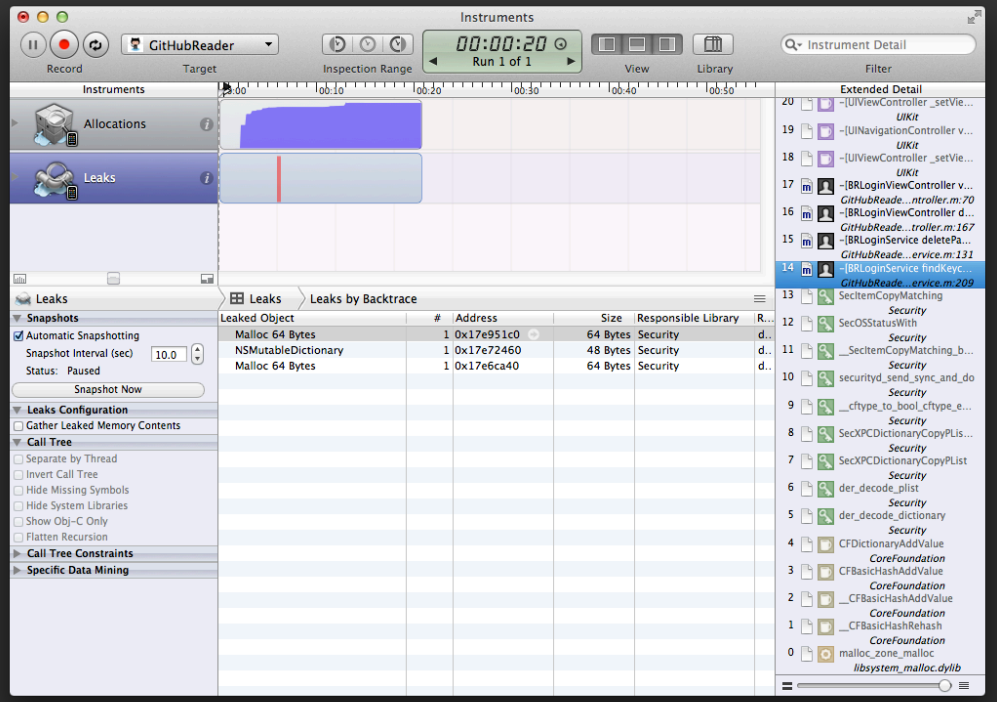
- Crash due to memory pressure.
- Slow app launch after being logged in.
- Choppy & blocking animation from parent to child views, particularly when opening a large list of commits.
- Choppy scrolling, particularly when a cell has images.

# What tends to be a sign of trouble?

- Memory Pressure
  - crashes
- CPU Pressure
  - slow UI rendering
  - choppiness
  - tight loops can cause memory pressure and/or crashes
- Network Overutilization
  - Lengthy or frequent waiting for next interaction
- Ignoring Perceived Performance
  - User panic can exacerbate the problem

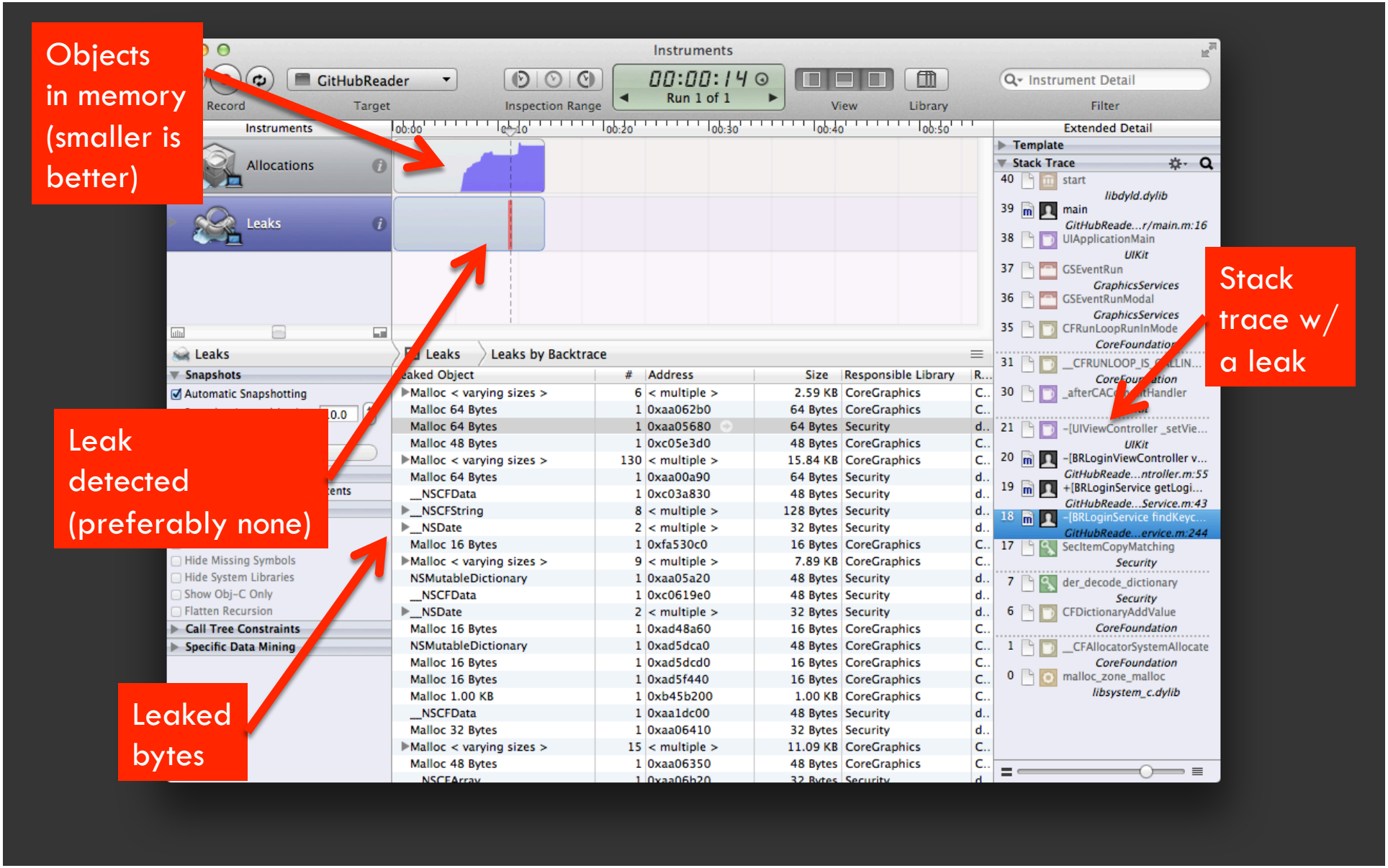
# Let's start analyzing with Instruments

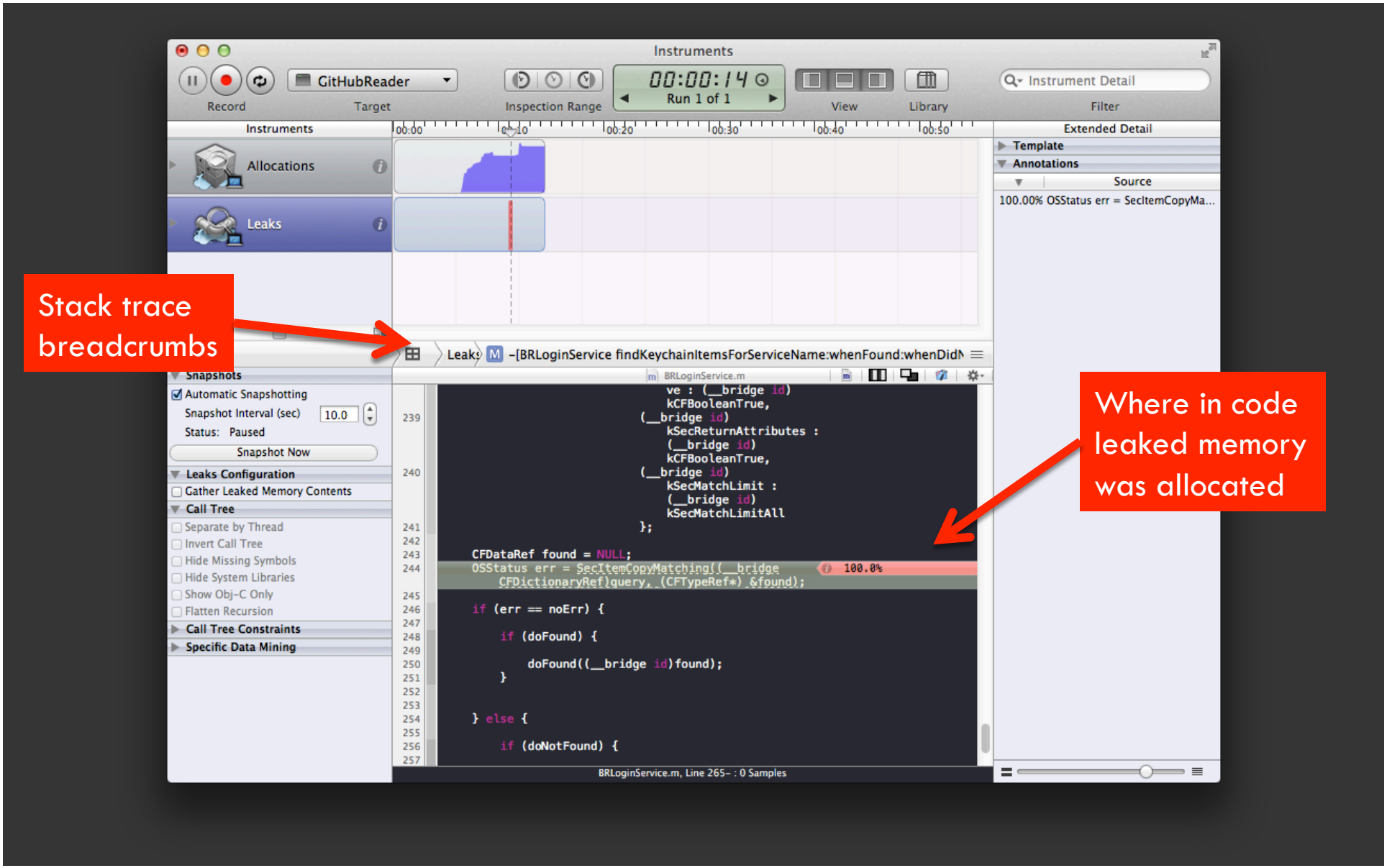
| | Device | Simulator |
|---|---|---|
| Leaks<br>memory leaks, blocking calls | ✓ | ✓ |
| Time Profiler<br>blocking calls | ✓ | ✓ |
| Activity Monitor<br>cpu, memory, network | ✓ | ✓ |
| Core Data<br>faults, reads, and writes | | ✓ |
| Core Animations<br>refresh rate | ✓ | |




# git checkout 01_perf_baseline

- Leaks Instruments
- Find the Leak


Objects in memory (smaller is better)
Leak detected (preferably none)
Leaked bytes
Stack trace w/ a leak
Instruments
GitHubReader
Record
Target
Inspection Range
00:00:14
Run 1 of 1
View
Library
Instrument Detail
Filter
Allocations
Leaks
Leaks by Backtrace
Snapshots
Automatic Snapshotting
Hide Missing Symbols
Hide System Libraries
Show Obj-C Only
Flatten Recursion
Call Tree Constraints
Specific Data Mining
Extended Detail
Template
Stack Trace


Instruments
Record
GitHubReader
Target
Inspection Range
00:00:14
Run 1 of 1
View
Library
Instrument Detail
Filter
Instruments
Allocations
Leaks
Extended Detail
Template
Annotations
Source
100.00% OSStatus err = SecItemCopyMa...
Leaks
-[BRLoginService findKeychainItemsForServiceName:whenFound:whenDidN
BRLoginService.m
Snapshots
Automatic Snapshotting
Snapshot Interval (sec)
10.0
Status: Paused
Snapshot Now
Leaks Configuration
Gather Leaked Memory Contents
Call Tree
Separate by Thread
Invert Call Tree
Hide Missing Symbols
Hide System Libraries
Show Obj-C Only
Flatten Recursion
Call Tree Constraints
Specific Data Mining
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
ve : (__bridge id)
kCFBooleanTrue,
(__bridge id)
kSecReturnAttributes :
(__bridge id)
kCFBooleanTrue,
(__bridge id)
kSecMatchLimit :
(__bridge id)
kSecMatchLimitAll
};
CFDataRef found = NULL;
OSStatus err = SecItemCopyMatching((__bridge
CFDictionaryRef)query, (CFTypeRef*) &found);
100.0%
if (err == noErr) {
if (doFound) {
doFound((__bridge id)found);
}
} else {
if (doNotFound) {
BRLoginService.m, Line 265– : 0 Samples
Stack trace breadcrumbs
Where in code leaked memory was allocated

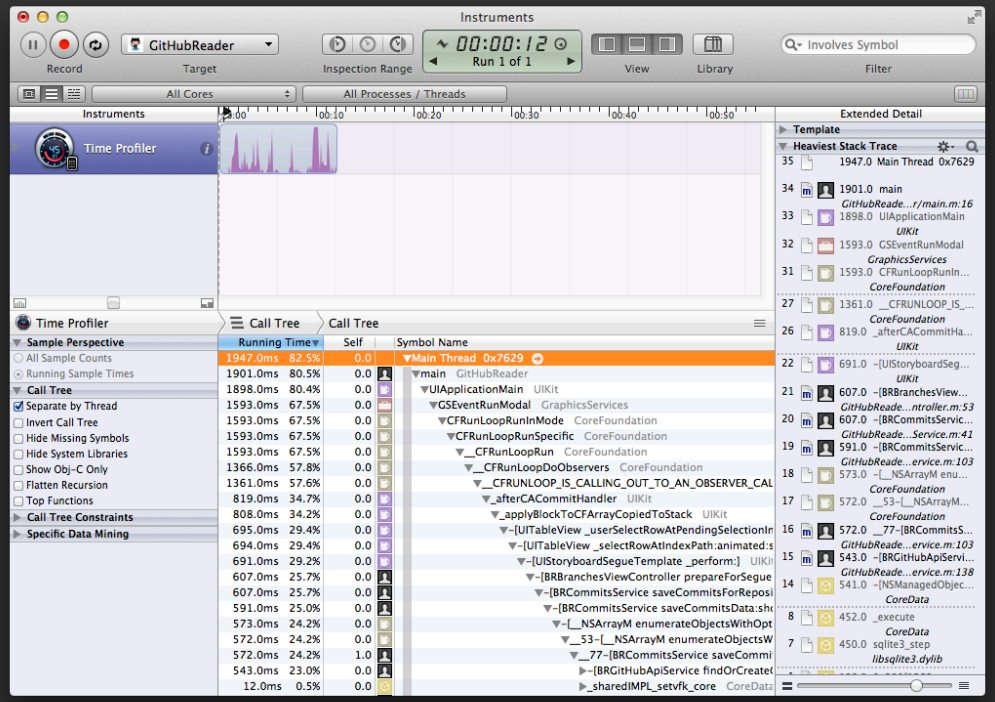
# git checkout 02_memory_leak

- ✓ Fix the memory leak

```
CFDataRef found = NULL;
OSStatus err = SecItemCopyMatching((__bridge CFDictionaryRef)query, (CFTypeRef*) &found);

if (err == noErr) {

            if (doFound) {

                        doFound((__bridge id)found);
            }

            CFRelease(found);

} else {

            if (doNotFound) {

                        doNotFound(query);
            }
}
```

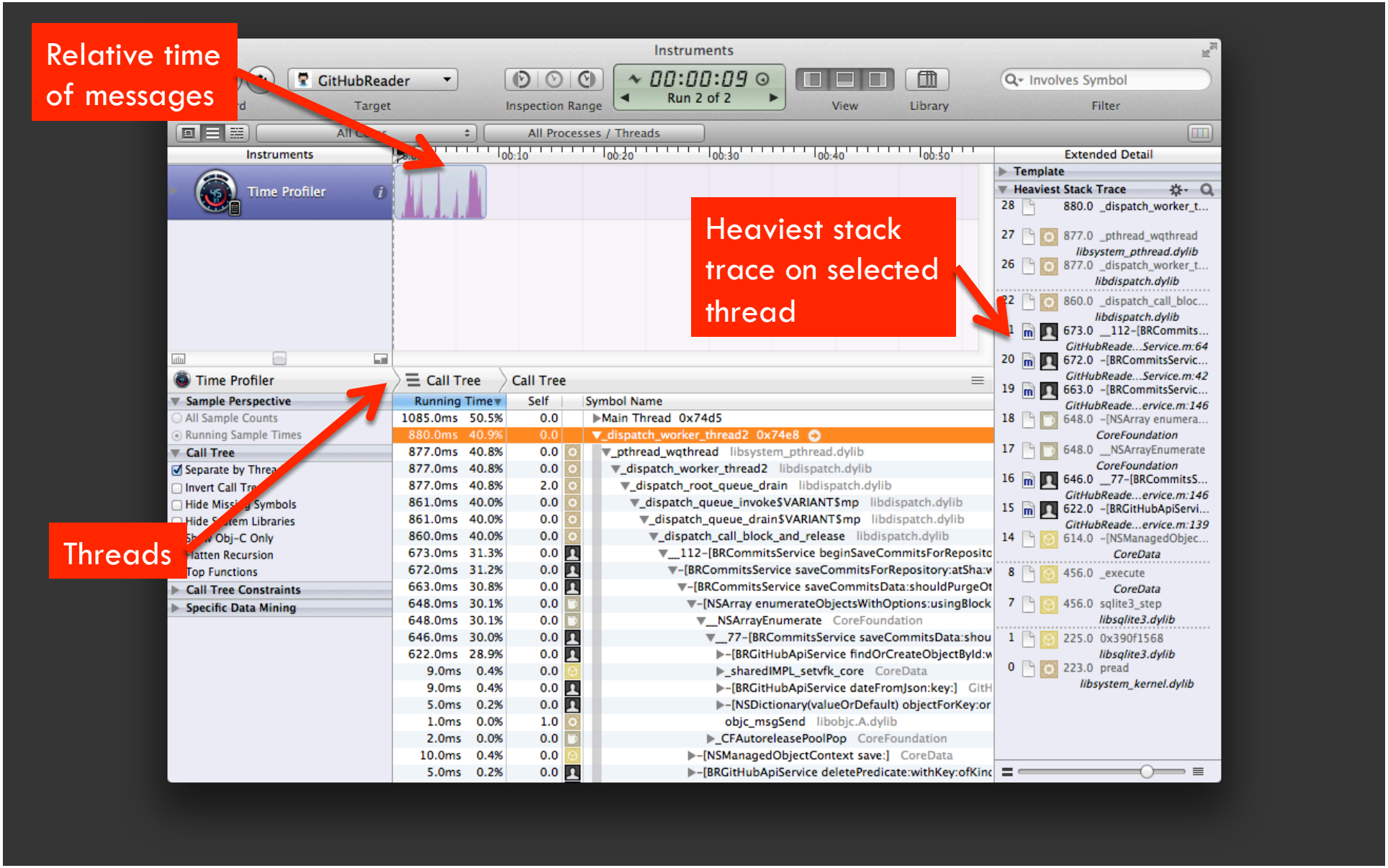
02_memory_leak

BRLoginService.m : 216




# git checkout 02_memory_leak

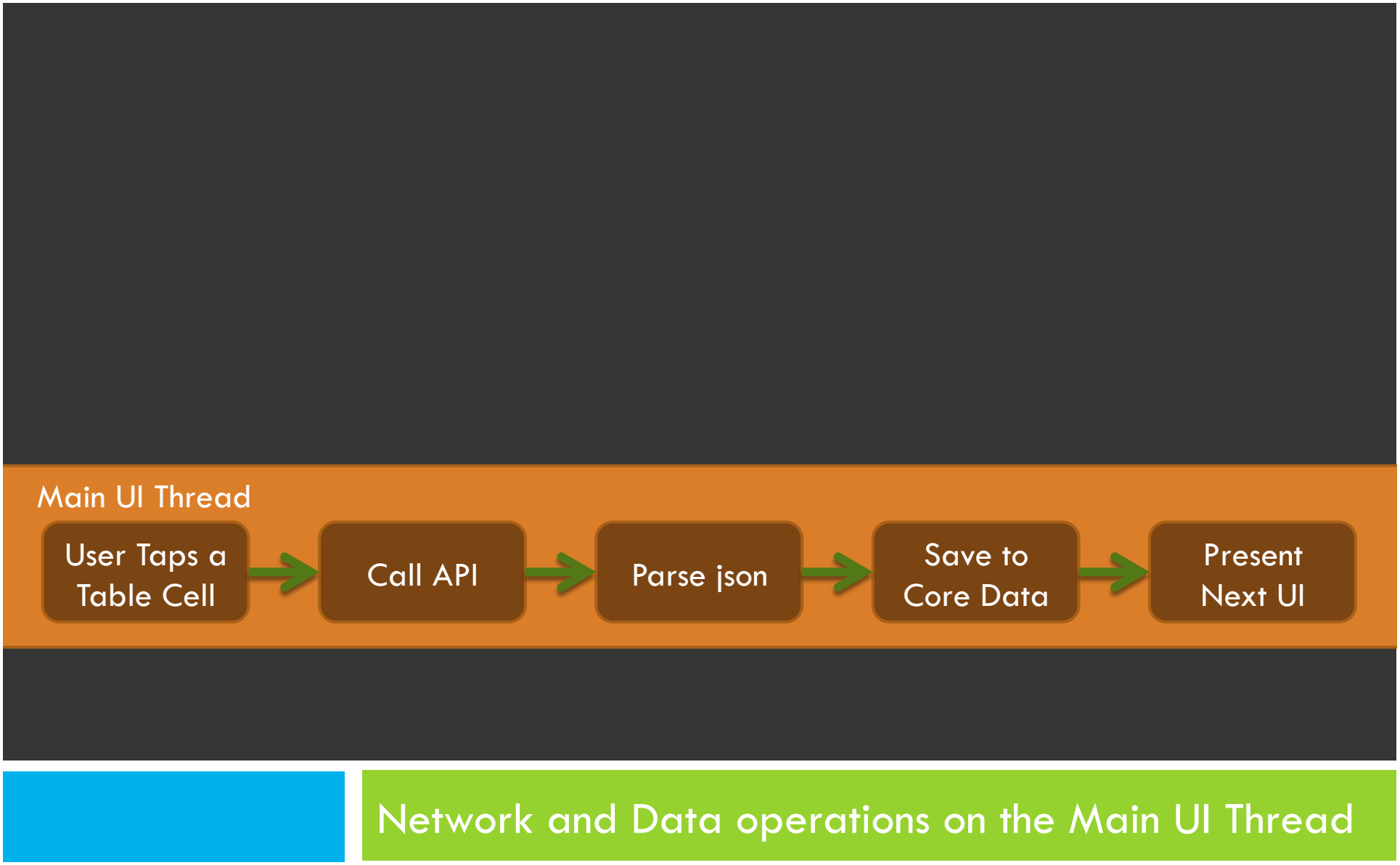
- Time Profiler Instruments
- Find the blocking messages
- Leaks Profiler Instruments
- Find the big data loop


Relative time of messages
Heaviest stack trace on selected thread
Threads
Instruments
GitHubReader
Target
Inspection Range
00:00:09
Run 2 of 2
View
Library
Involves Symbol
Filter
All Processes / Threads
Time Profiler
Extended Detail
Template
Heaviest Stack Trace
Call Tree
Sample Perspective
All Sample Counts
Running Sample Times
Call Tree
Separate by Thread
Invert Call Tree
Hide Missing Symbols
Hide System Libraries
Show Obj-C Only
Flatten Recursion
Top Functions
Call Tree Constraints
Specific Data Mining
Running Time
Self
Symbol Name
1085.0ms 50.5% 0.0 Main Thread 0x74d5
880.0ms 40.9% 0.0 _dispatch_worker_thread2 0x74e8
877.0ms 40.8% 0.0 _pthread_wqthread libsystem_pthread.dylib
877.0ms 40.8% 0.0 _dispatch_worker_thread2 libdispatch.dylib
877.0ms 40.8% 2.0 _dispatch_root_queue_drain libdispatch.dylib
861.0ms 40.0% 0.0 _dispatch_queue_invoke$VARIANT$mp libdispatch.dylib
861.0ms 40.0% 0.0 _dispatch_queue_drain$VARIANT$mp libdispatch.dylib
860.0ms 40.0% 0.0 _dispatch_call_block_and_release libdispatch.dylib
673.0ms 31.3% 0.0 __112-[BRCommitsService beginSaveCommitsForReposito
672.0ms 31.2% 0.0 -[BRCommitsService saveCommitsForRepository:atSha:v
663.0ms 30.8% 0.0 -[BRCommitsService saveCommitsData:shouldPurgeOt
648.0ms 30.1% 0.0 -[NSArray enumerateObjectsWithOptions:usingBlock
648.0ms 30.1% 0.0 __NSArrayEnumerate CoreFoundation
646.0ms 30.0% 0.0 __77-[BRCommitsService saveCommitsData:shou
622.0ms 28.9% 0.0 -[BRGitHubApiService findOrCreateObjectById:w
9.0ms 0.4% 0.0 _sharedIMPL_setvfk_core CoreData
9.0ms 0.4% 0.0 -[BRGitHubApiService dateFromJson:key:] GitH
5.0ms 0.2% 0.0 -[NSDictionary(valueOrDefault) objectForKey:or
1.0ms 0.0% 1.0 objc_msgSend libobjc.A.dylib
2.0ms 0.0% 0.0 _CFAutoreleasePoolPop CoreFoundation
10.0ms 0.4% 0.0 -[NSManagedObjectContext save:] CoreData
5.0ms 0.2% 0.0 -[BRGitHubApiService deletePredicate:withKey:ofKinc
28 880.0 _dispatch_worker_t...
27 877.0 _pthread_wqthread
libsystem_pthread.dylib
26 877.0 _dispatch_worker_t...
libdispatch.dylib
22 860.0 _dispatch_call_bloc...
libdispatch.dylib
673.0 __112-[BRCommits...
GitHubReade...Service.m:64
20 672.0 -[BRCommitsServic...
GitHubReade...Service.m:42
19 663.0 -[BRCommitsServic...
GitHubReade...ervice.m:146
18 648.0 -[NSArray enumera...
CoreFoundation
17 648.0 __NSArrayEnumerate
CoreFoundation
16 646.0 __77-[BRCommitsS...
GitHubReade...ervice.m:146
15 622.0 -[BRGitHubApiServi...
GitHubReade...ervice.m:139
14 614.0 -[NSManagedObjec...
CoreData
8 456.0 _execute
CoreData
7 456.0 sqlite3_step
libsqlite3.dylib
1 225.0 0x390f1568
libsqlite3.dylib
0 223.0 pread
libsystem_kernel.dylib

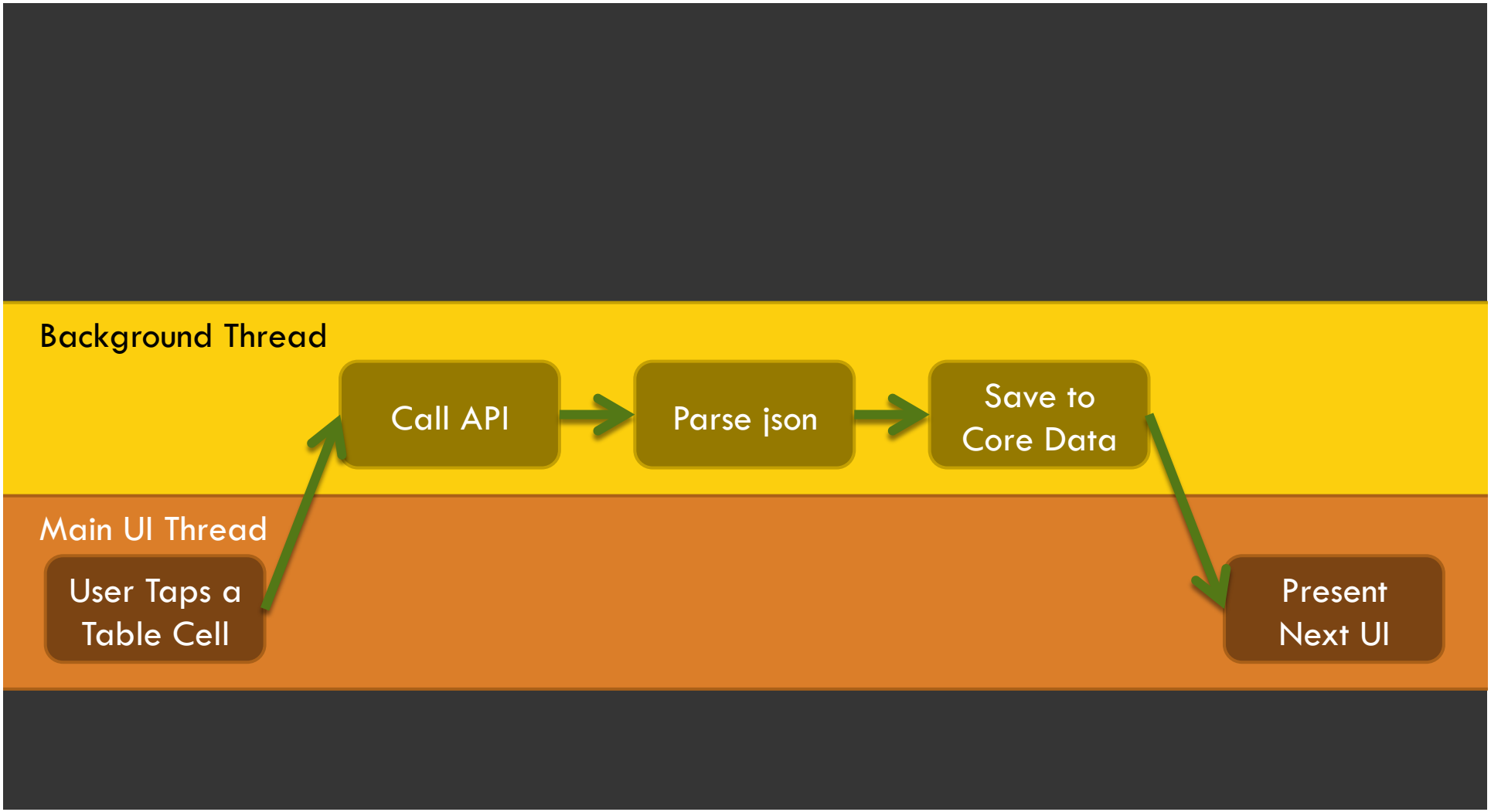
# Network access and data manipulation blocking the main UI thread

# Do as little as possible on the Main UI Thread

- Not a simple task
- Several things that should be done off the main thread
  - Retrieve data from API
  - Parse it (deserialize json, spin through arrays, etc.)
  - Store it in Core Data
  - Communicate completion or failure back to the main UI thread




Network and Data operations on the Main UI Thread




Network and Data operations on the Main UI Thread

# Asyncronous Network & Data Options

- Basic implementation of id<NSURLConnectionDataDelegate>
  - Pros
    - Network activity happens asyncronously
    - Can cancel a connection
    - Relatively simple, boilerplate code
    - Can choose how to store incoming data chunks (in memory, page to disk, etc.)
  - Cons
    - In simplest implementation, delegate callbacks come back to main UI thread

# Asyncronous Network & Data Options

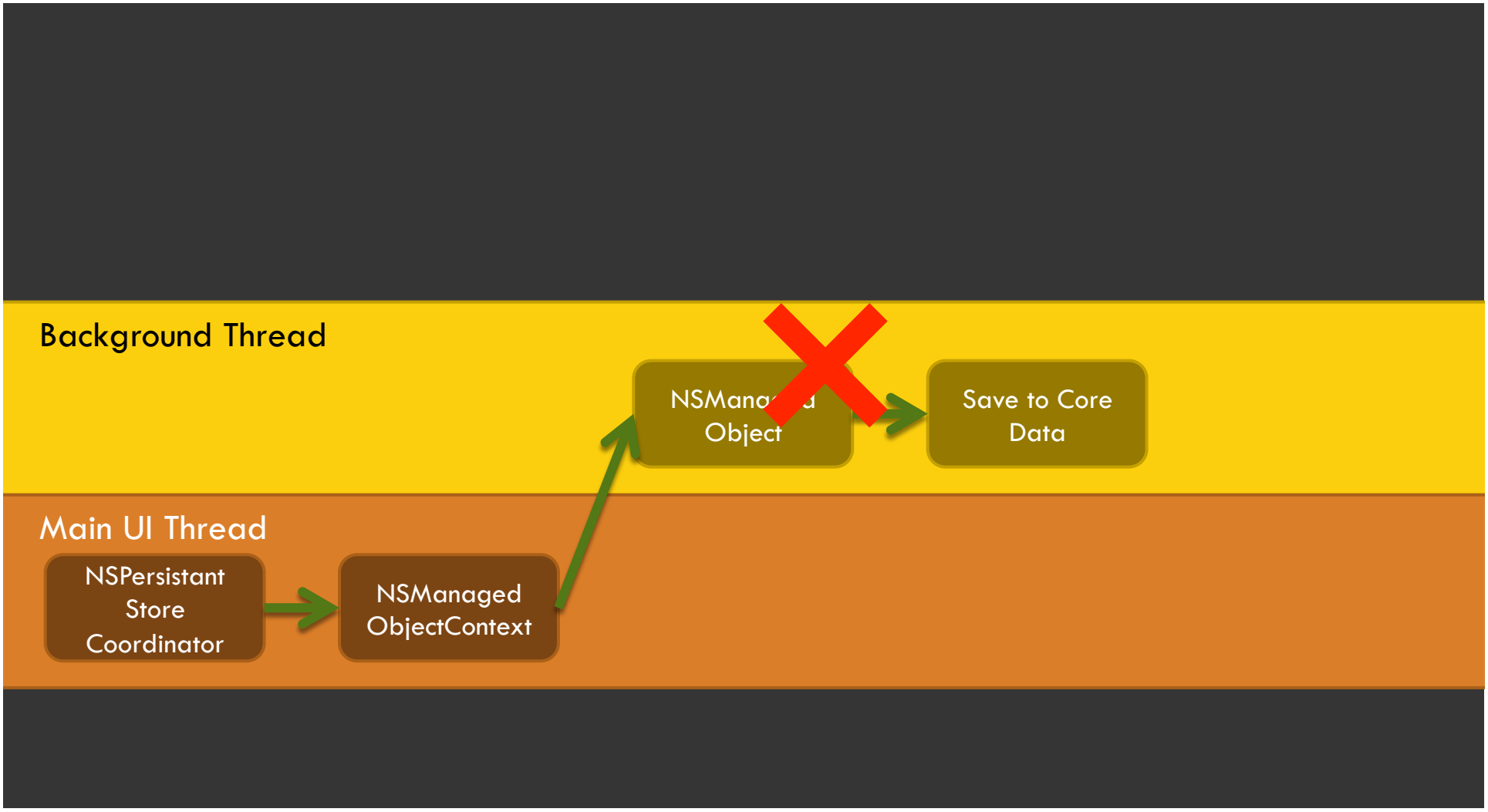
- NSOperation and NSOperationQueue
  - Pros
    - Can enqueue multiple operations and throttle concurrency
    - Can create rich dependancies between operations
  - Cons
    - Must handle for Core Data concurrency rules

# Asyncronous Network & Data Options

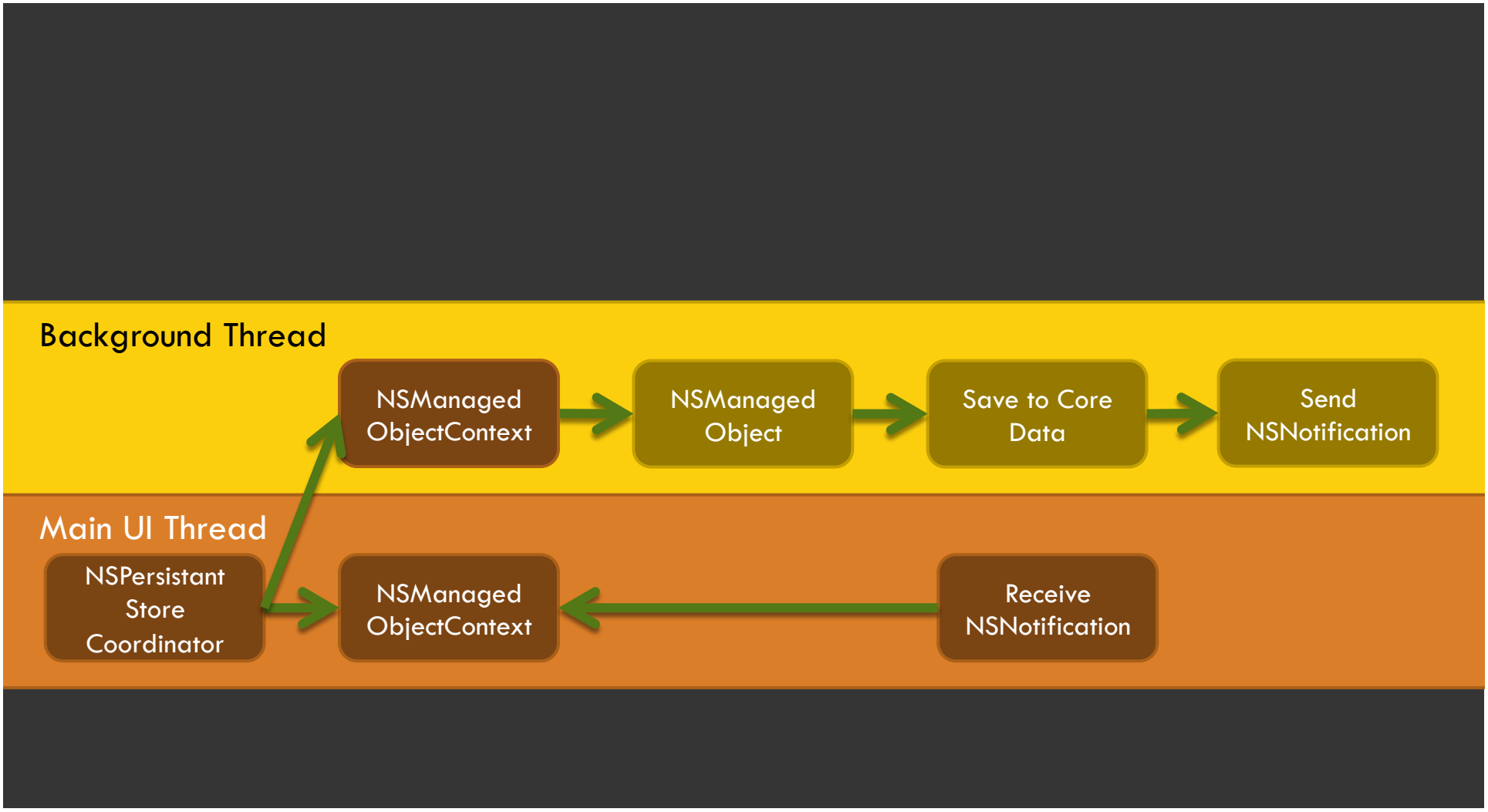
- Grand Central Dispatch
  - Pros
    - Very easy to port syncronous calls to background thread
  - Cons
    - Must handle for Core Data concurrency rules

# Core Data concurrency rules

- NSPersistantStoreCoordinator is thread safe
- NSManagedObjectContext is not thread safe
- NSManagedObjectID is thread safe
- NSManagedObject is not thread safe




Network and Data operations on the Main UI Thread




Network and Data operations on the Main UI Thread

# git checkout 03_async_gcd

- ✓ Async network and Core Data using GCD
- ✓ Paging large data sets
- ✓ Turn off Gravatars for the moment

# git checkout 03_async_gcd

- ✓ Async network and Core Data using GCD

```
- (BOOL)saveDataWithError:(NSError **)error {

        // Do this work on the main UI thread
        NSArray *json = [self getRemoteData];
        NSManagedObjectContext *context = [[BRModelManager sharedInstance] context];

        [json enumerateObjectsUsingBlock:^(id obj, NSUInteger idx, BOOL *stop) {

        }];

        return [context save:error];
}
```

02_memory_leak

BROrganizationService.m

```objc
- (void)beginSaveDataWithCompletion:(void (^)(BOOL saved, NSError *error))completion {
            dispatch_queue_t serviceQueue = dispatch_queue_create("Service queue", NULL);
            dispatch_async(serviceQueue, ^{

                        // do this work off the main ui thread
                        NSArray *json = [self getRemoteData];
                        NSManagedObjectContext *context = [[NSManagedObjectContext alloc] init];
                        [context setPersistentStoreCoordinator:
                         [[BRModelManager sharedInstance] persistentStoreCoordinator]];

                        [json enumerateObjectsUsingBlock:^(id obj, NSUInteger idx, BOOL *stop) {

                        }];

                        dispatch_async(dispatch_get_main_queue(), ^{

                                    // do this work on the main ui thread
                                    completion(answer, error);
                        });
            });
}
```

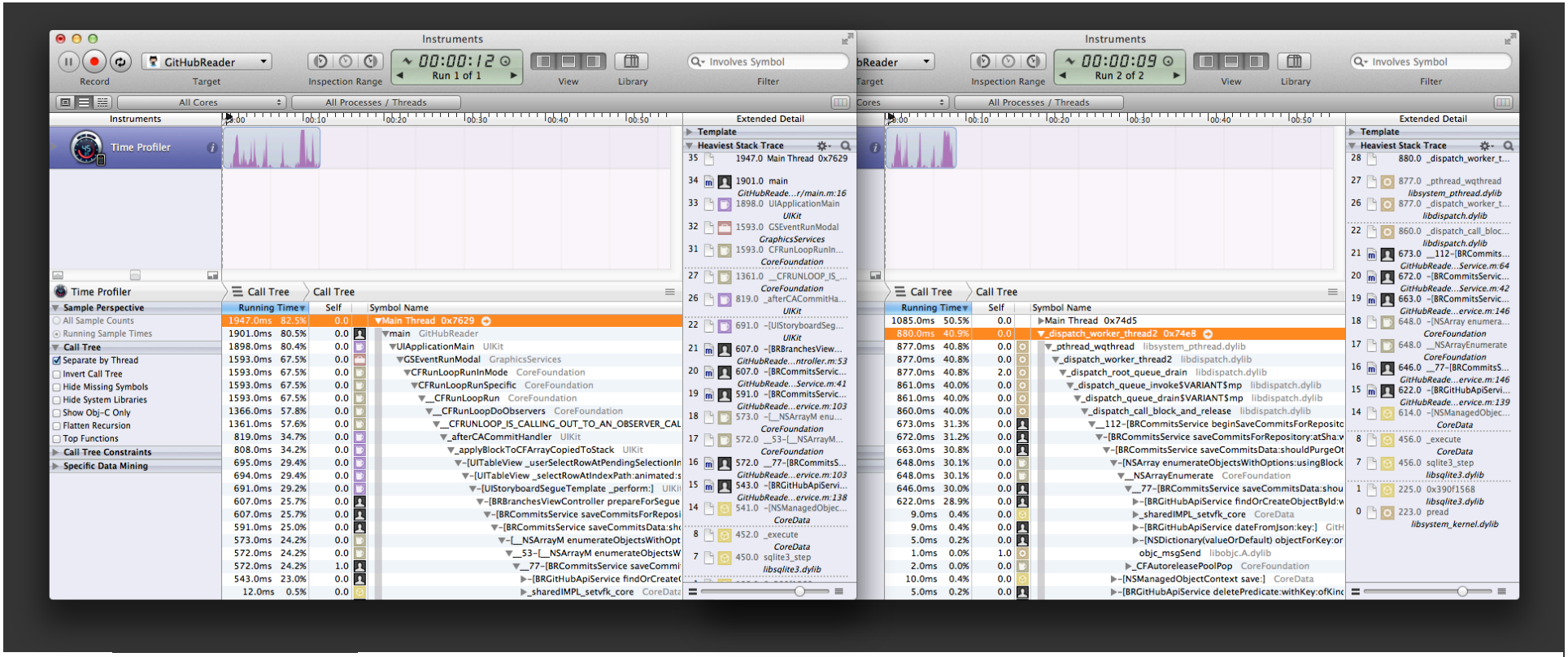
03_async_gcd

BROrganizationService.m

```objc
- (IBAction)didTapButton:(id)sender {

        NSError *error = nil;
        Service *service = [[Service alloc] init];
        if (![service doSomeDataOperation error:&error]) return;

        // Update the UI on the main thread
}
```

```objc
- (IBAction)didTapButton:(id)sender {

        Service *service = [[Service alloc] init];
        [service beginSomeDataOperation withCompletion:^(BOOL saved, NSError *error) {

                // Update the UI on the main thread
        }];
}
```

BROrganizationsViewController.m




# git checkout 03_async_gcd

- Time Profiler Instruments to check progress
- Most of “our” work is now happening off the main thread

# git checkout 03_async_gcd

- ✓ Paging large data sets

```
- (BOOL)saveCommitsWithError:(NSError **)error {

        NSError* inError = nil;
        NSMutableArray *all = [NSMutableArray arrayWithCapacity:0];
        while (sha) {

                NSArray *json = [self getRemoteDataWithError:&inError];
                if (!json || inError) {

                        *error = inError;
                        return NO;
                }
                sha = [[json lastObject] valueForKeyPath:@"parents.@max.sha"];
                [all addObjectsFromArray:json];
        }

        return [self saveCommits:all withError:error];
}
```

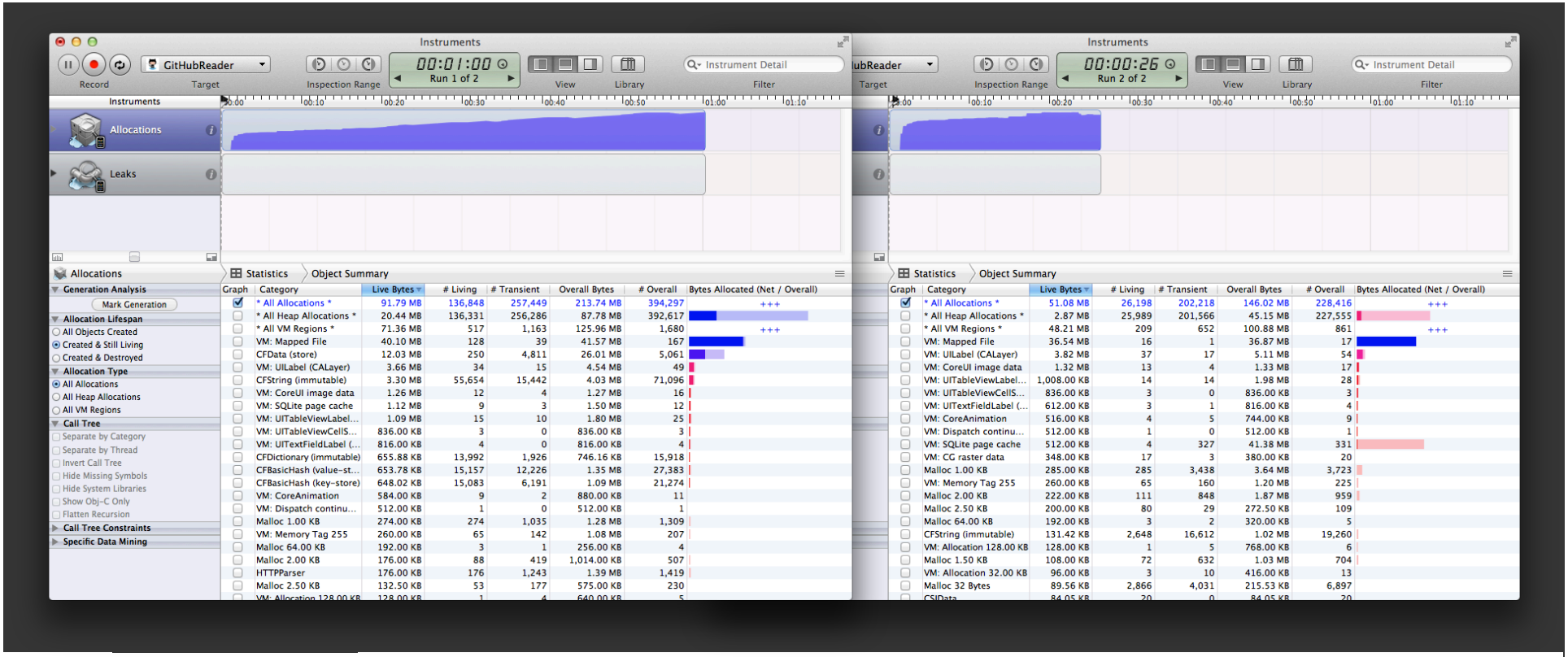
02_memory_leak

BRCommitsService.m

```objc
- (BOOL)saveCommitsWithError:(NSError **)error {

        NSError *inError = nil;
        NSArray *json = [self getRemoteDataWithError:&inError];
        if (!json || inError) {

                *error = inError;
                return NO;
        }

        return [self saveCommits:json withError:error];
}
```

03_async_gcd

BRCommitsService.m




# git checkout 03_async_gcd

- Leaks Instruments to check progress
- Much less memory pressure
- Must faster view of first page of commits

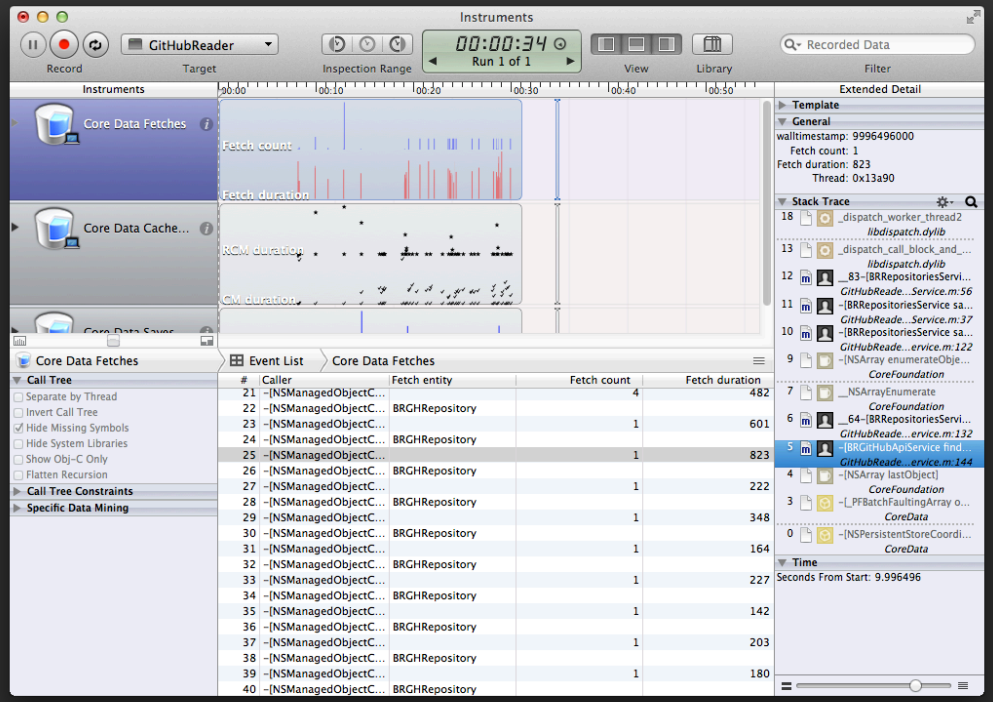
# git checkout 04_prefetch

- ✓ Batch fetch for updates

```objc
NSFetchRequest *fetchRequest = [[NSFetchRequest alloc] init];
        [fetchRequest setRelationshipKeyPathsForPrefetching:@[@"author"]];
        [fetchRequest setFetchBatchSize:20];
```

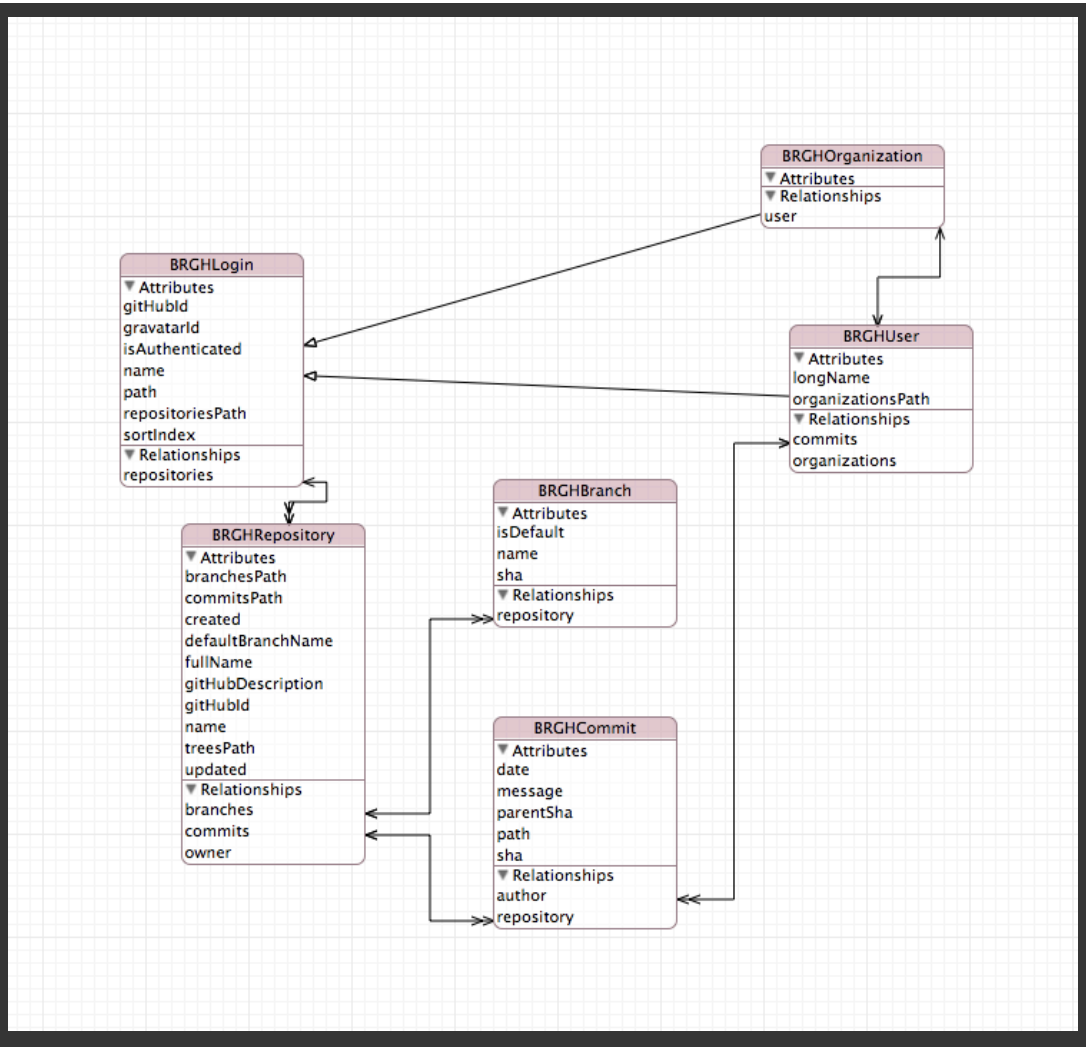
04_prefetch

BRCommitsViewController.m



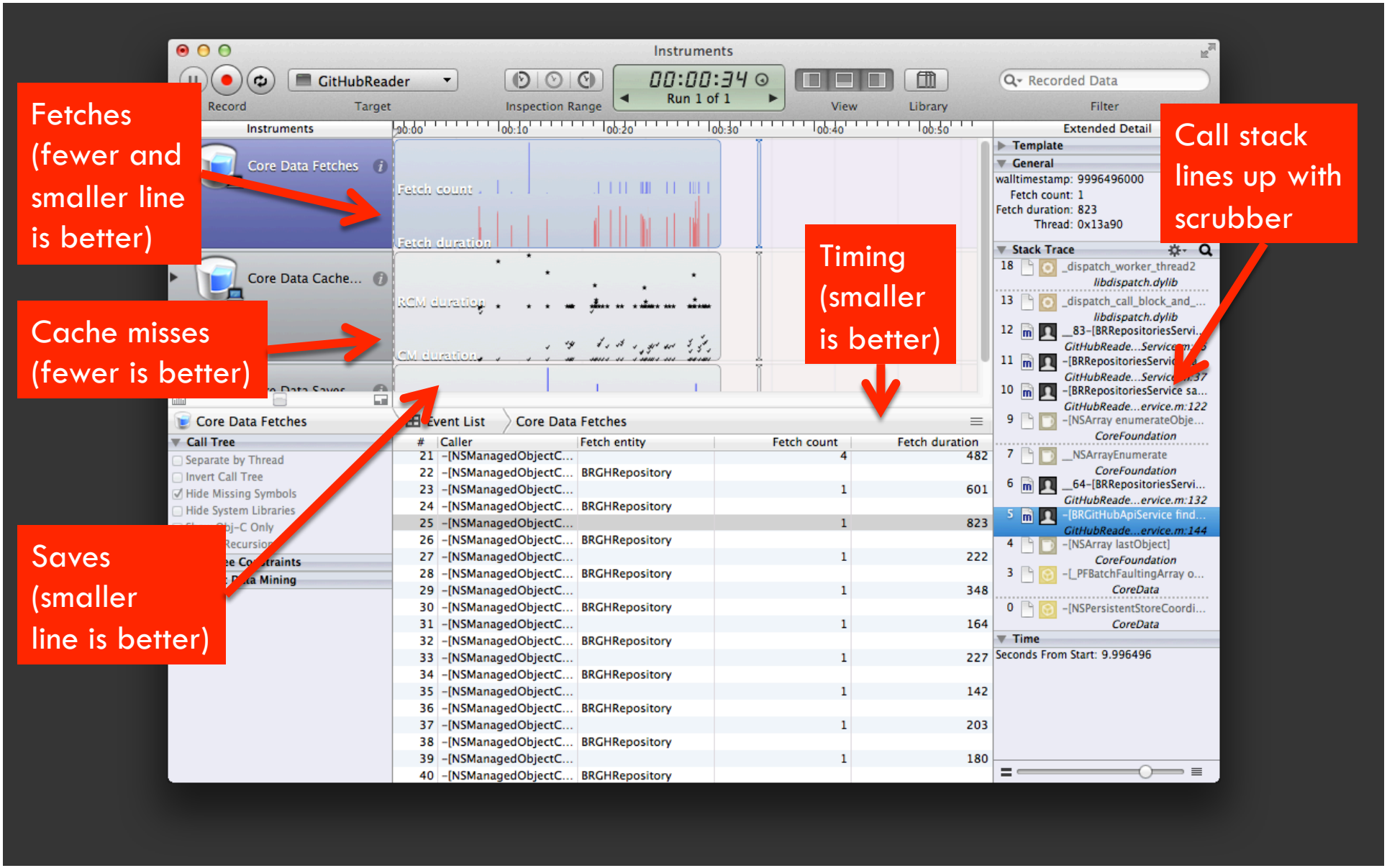

# git checkout 04_prefetch

- CoreData Instruments
- Only works on Simulator
- Lots of fetches
- Lots of faults




# 03_async_gcd

# GitHubReader.xcdatamodel


Fetches (fewer and smaller line is better)
Cache misses (fewer is better)
Saves (smaller line is better)
Timing (smaller is better)
Call stack lines up with scrubber
Instruments
GitHubReader
Record
Target
Inspection Range
00:00:34
Run 1 of 1
View
Library
Recorded Data
Filter
Core Data Fetches
Core Data Cache...
Fetch count
Fetch duration
RCM duration
CM duration
Event List
Core Data Fetches
Call Tree
Separate by Thread
Invert Call Tree
Hide Missing Symbols
Hide System Libraries
Extended Detail
Template
General
walltimestamp: 9996496000
Fetch count: 1
Fetch duration: 823
Thread: 0x13a90
Stack Trace
Time
Seconds From Start: 9.996496
# Caller Fetch entity Fetch count Fetch duration
21 -[NSManagedObjectC... 4 482
22 -[NSManagedObjectC... BRGHRepository
23 -[NSManagedObjectC... 1 601
24 -[NSManagedObjectC... BRGHRepository
25 -[NSManagedObjectC... 1 823
26 -[NSManagedObjectC... BRGHRepository
27 -[NSManagedObjectC... 1 222
28 -[NSManagedObjectC... BRGHRepository
29 -[NSManagedObjectC... 1 348
30 -[NSManagedObjectC... BRGHRepository
31 -[NSManagedObjectC... 1 164
32 -[NSManagedObjectC... BRGHRepository
33 -[NSManagedObjectC... 1 227
34 -[NSManagedObjectC... BRGHRepository
35 -[NSManagedObjectC... 1 142
36 -[NSManagedObjectC... BRGHRepository
37 -[NSManagedObjectC... 1 203
38 -[NSManagedObjectC... BRGHRepository
39 -[NSManagedObjectC... 1 180
40 -[NSManagedObjectC... BRGHRepository

```
- (BOOL)saveData:(NSArray *)json error:(NSError **)error {

        // context set up
        [json enumerateObjectsUsingBlock:^(id obj, NSUInteger idx, BOOL *stop) {

                NSDictionary *itemJson = (NSDictionary *)obj;
                BRGHCommit *commit = (BRGHCommit *)[apiService findOrCreateObjectById:sha];
                [commit setMessage:[itemJson objectForKey:@"commit.message" orDefault:nil]];
                // more reading json
        }];

        // relationships and cleanup
        return [_context save:error];
}
```

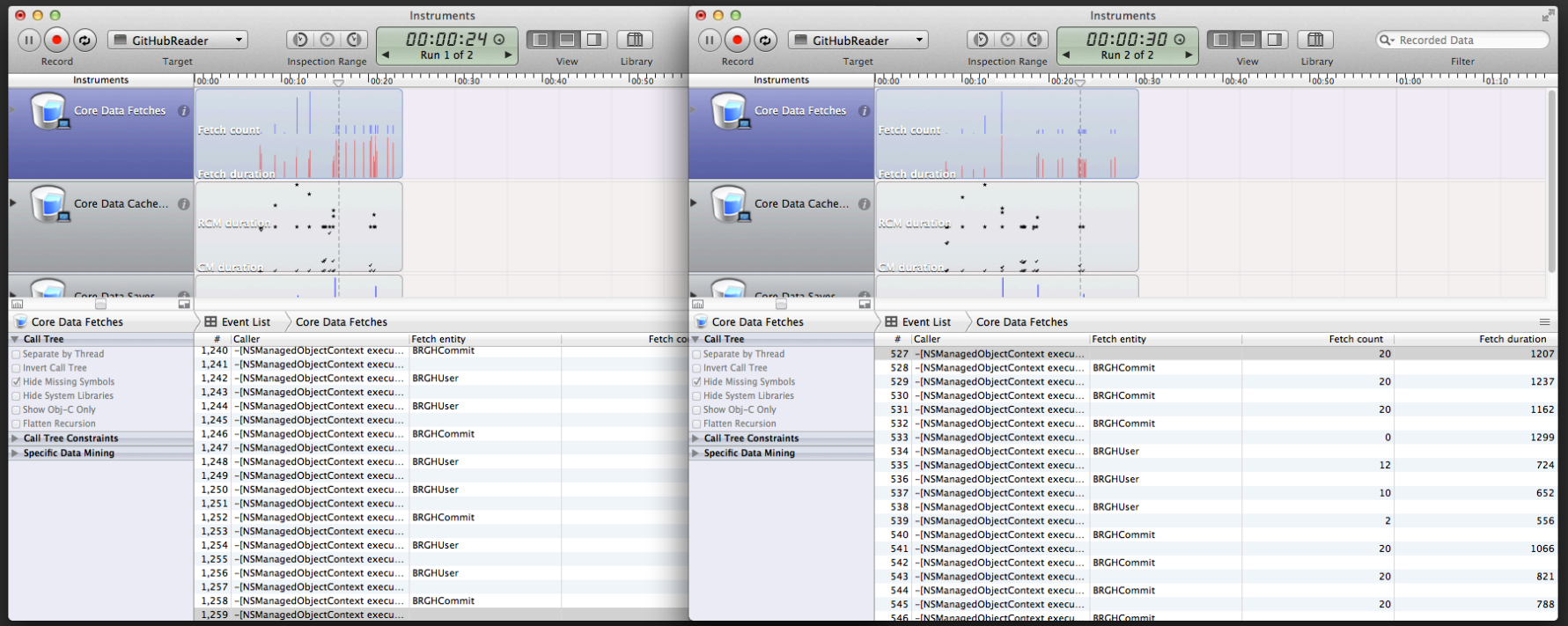
04_prefetch

BRCommitsService.m

```
- (BOOL)saveData:(NSArray *)json error:(NSError **)error {
        // context set up
        // fetch all soon-to-be updated commits
        NSArray *shas = [json valueForKeyPath:@"@distinctUnionOfObjects.sha"];
        NSSortDescriptor *shaSort = [NSSortDescriptor sortDescriptorWithKey:key ascending:YES];
        NSArray *oldCommitsArray = [apiService findObjectsByIds:shas];

        [json enumerateObjectsUsingBlock:^(id obj, NSUInteger idx, BOOL *stop) {

                NSDictionary *itemJson = (NSDictionary *)obj;
                NSString *sha = (NSString *)[itemJson valueForKeyPath:@"sha"];
                BRGHCommit *commit = oldCommits[sha];
                if (!commit) {
                        commit = [NSEntityDescription insertNewObject...];
                        [commit setSha:sha];
                }
                // ... proceed forward as before
```

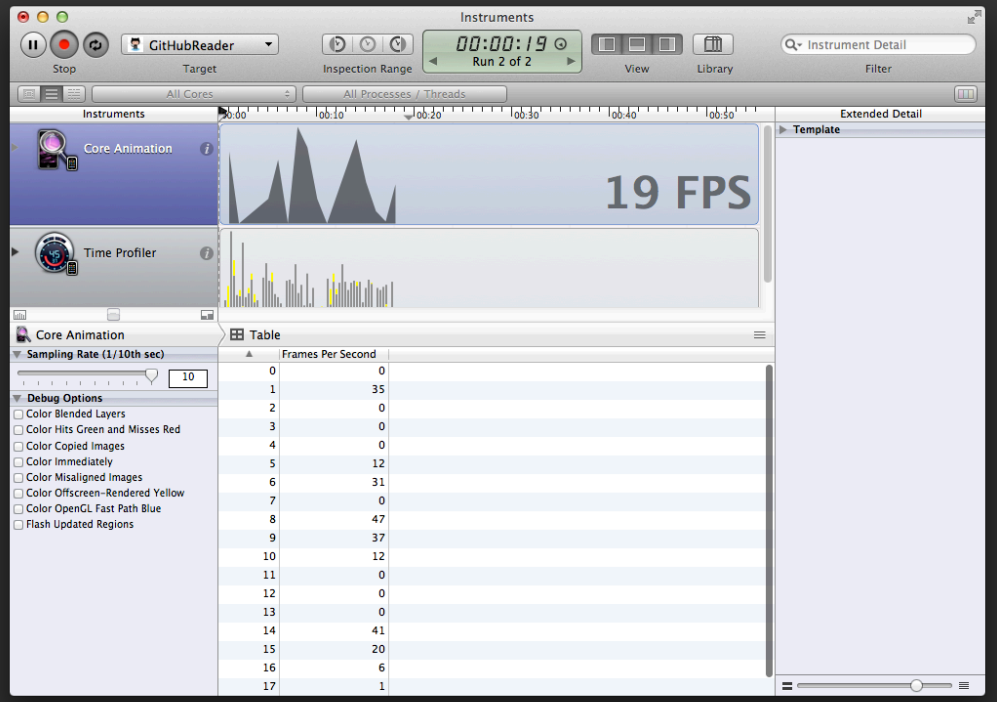
05_batch_fetch_update

BRCommitsService.m



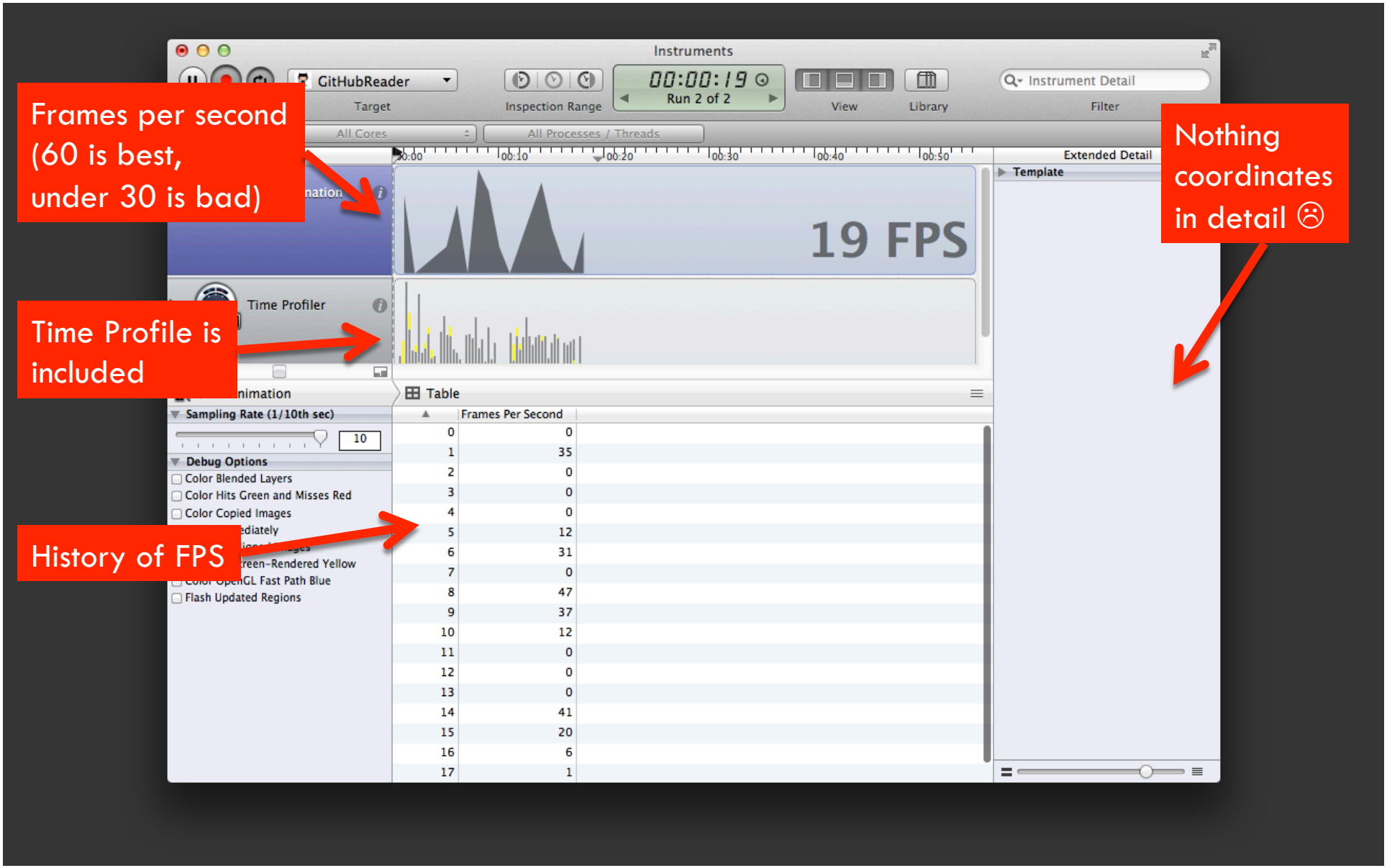

# git checkout 05_batch_fetch_update

- Core Data Instruments to check progress
- One-off fetches for updates now batched




# git checkout 02_memory_leak

- Core Animation Instruments
- Only works on Device
- ( reset app on device )
- Slow table scrolling


Frames per second (60 is best, under 30 is bad)
Time Profile is included
History of FPS
Nothing coordinates in detail ☹
Instruments
GitHubReader
Target
Inspection Range
00:00:19
Run 2 of 2
View
Library
Instrument Detail
Filter
All Cores
All Processes / Threads
19 FPS
Time Profiler
Extended Detail
Template
Table
Sampling Rate (1/10th sec)
10
Debug Options
Color Blended Layers
Color Hits Green and Misses Red
Color Copied Images
Flash Updated Regions
Frames Per Second
0 0
1 35
2 0
3 0
4 0
5 12
6 31
7 0
8 47
9 37
10 12
11 0
12 0
13 0
14 41
15 20
16 6
17 1

# git checkout 06_caching

- ✓ HTTP LastModified
- ✓ Local NSCache

```
- (void)configureCell:(UITableViewCell *)cell atIndexPath:(NSIndexPath *)indexPath {

        BRGHCommit *commit = (BRGHCommit *)_commits[indexPath.row];
        NSString *gravatarId = commit.author.gravatarId;
        UIImage *image = [BRGravatarService imageForGravatarWithHash:gravatarId ofSize:80];
        [cell.imageView setImage:image];
        [cell.textLabel setText:commit.message];
}
```

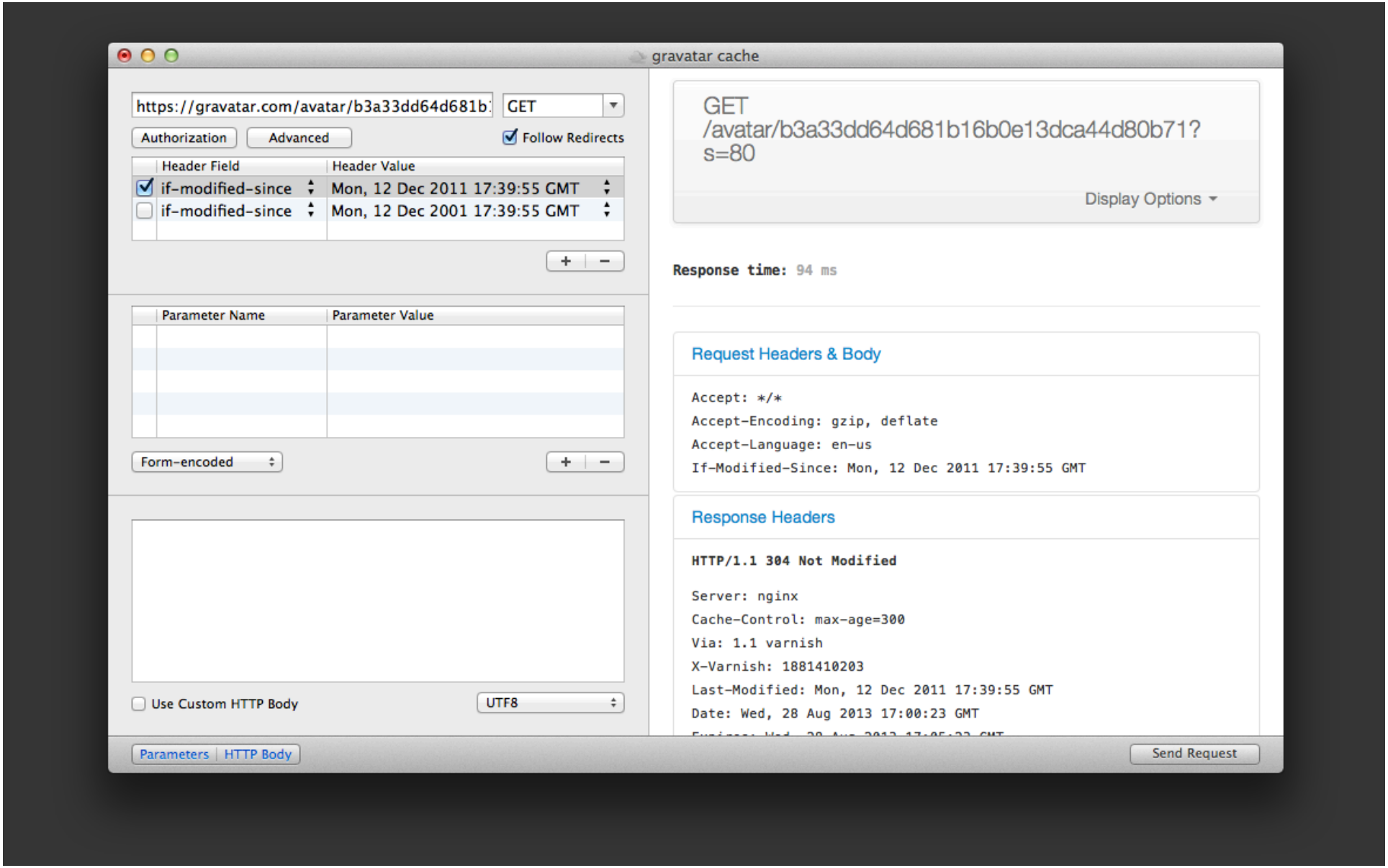
02_memory_leak

BRCommitsViewController.m

```
+ (UIImage *)imageForGravatarWithHash:(NSString *)hash ofSize:(int)size {

        [[UIApplication sharedApplication] setNetworkActivityIndicatorVisible:YES];
        NSURL *url = [self urlForGravatarWithHash:hash ofSize:size];
        UIImage *gravatarDownload = [UIImage imageWithData:[NSData dataWithContentsOfURL:url]];
        UIImage *answer = [UIImage imageWithCGImage:[gravatarDownload CGImage]
          scale:[[UIScreen mainScreen] scale]
        orientation:UIImageOrientationUp];

        [[UIApplication sharedApplication] setNetworkActivityIndicatorVisible:NO];

        return answer;
}
```

02_memory_leak

BRGravatarService.m


gravatar cache
https://gravatar.com/avatar/b3a33dd64d681b
GET
Authorization
Advanced
Follow Redirects
Header Field
Header Value
if-modified-since
Mon, 12 Dec 2011 17:39:55 GMT
if-modified-since
Mon, 12 Dec 2001 17:39:55 GMT
Parameter Name
Parameter Value
Form-encoded
Use Custom HTTP Body
UTF8
Parameters | HTTP Body
GET /avatar/b3a33dd64d681b16b0e13dca44d80b71?s=80
Display Options
Response time: 94 ms
Request Headers & Body
Accept: */*
Accept-Encoding: gzip, deflate
Accept-Language: en-us
If-Modified-Since: Mon, 12 Dec 2011 17:39:55 GMT
Response Headers
HTTP/1.1 304 Not Modified
Server: nginx
Cache-Control: max-age=300
Via: 1.1 varnish
X-Varnish: 1881410203
Last-Modified: Mon, 12 Dec 2011 17:39:55 GMT
Date: Wed, 28 Aug 2013 17:00:23 GMT
Send Request

```
- (void)configureCell:(UITableViewCell *)cell atIndexPath:(NSIndexPath *)indexPath {

        BRGHCommit *commit = (BRGHCommit *)_commits[indexPath.row];
        UIImage *image = [_gravatarService cachedImageForLogin:commit.author];
        UIImage *placeImage = image
        ? image
        : [UIImage imageNamed:@"cell60"];
        [cell.imageView setImage:placeImage];
}
```

06_caching

BRCommitsViewController.m

```
- (id)init {

        self = [super init];
        if (self) {

                NSOperationQueue *queue = [[NSOperationQueue alloc] init];
                [queue setName:@"BRGravatarService queue"];
                [queue setMaxConcurrentOperationCount:3];
                _queue = queue;

                _thumbnailCache = [[NSCache alloc] init];
        }

        return self;
}
```

06_caching

BRGravatarService.m

```
- (void)saveGravatarsForLogin:(BRGHLogin *)login ofSize:(int)size {
          NSMutableURLRequest *request = [[NSMutableURLRequest alloc] ...
          [request setValue:lastModified forHTTPHeaderField:@"if-modified-since"];
          [NSURLConnection sendAsynchronousRequest:request queue:_queue
                    completionHandler:^(NSURLResponse *response, NSData *data, NSError
                    *connectionError) {[gravatar setImage:data];


                    [gravatar setLastModified:httpResponse.allHeaderFields[@"Last-Modified"]];
                    [context save:NULL];
          }];
}
```

06_caching

BRGravatarService.m


# git checkout 06_caching

- Core Animation Instruments to check progress
- Fetching images off main UI thread
- Using HTTP caching and local caching

# git checkout 06_caching

- ✓ This works for API calls too if the API vendor implements it

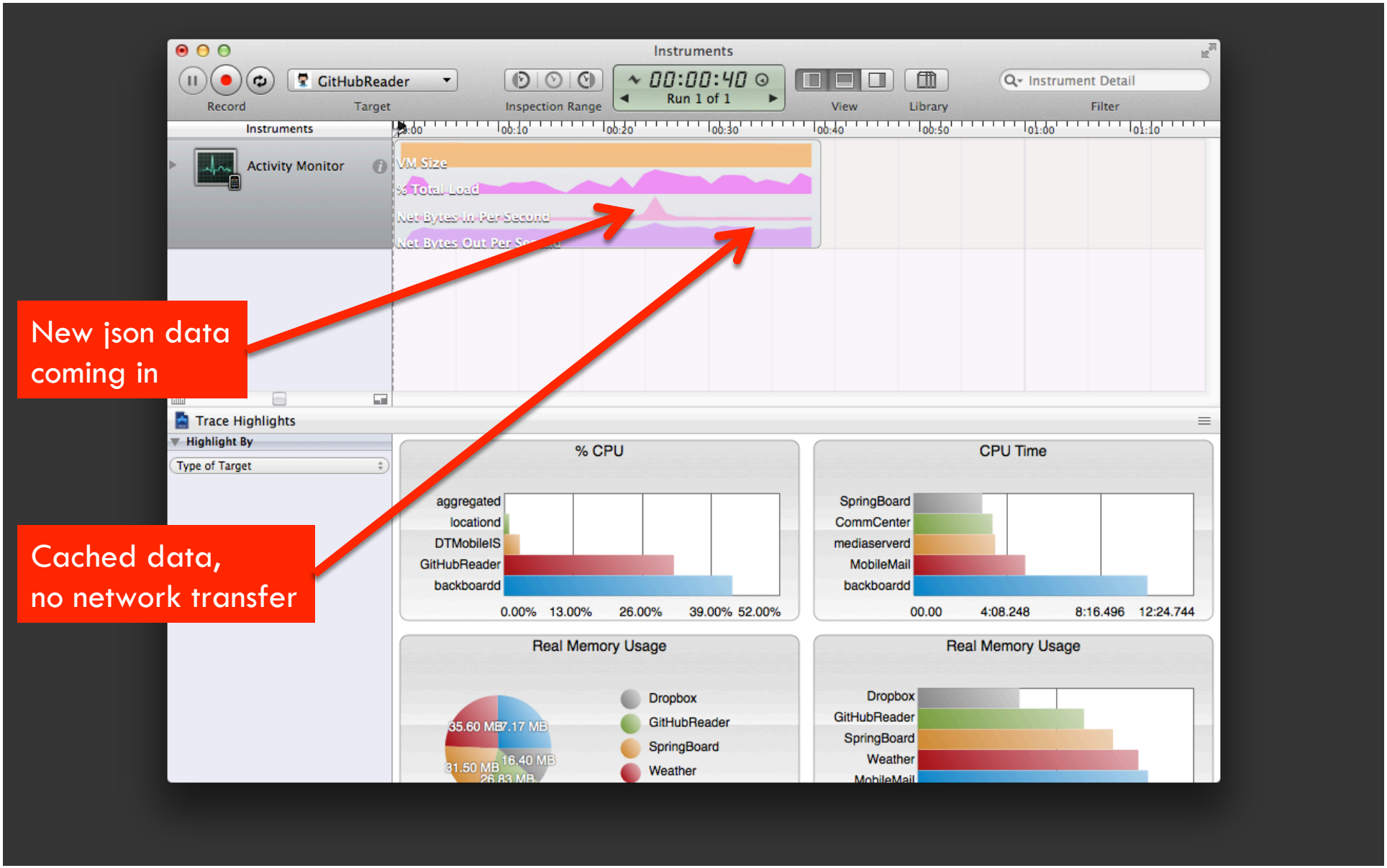
```
- (NSArray *)getRemoteData lastModified:(NSString **)lastModified withError:(NSError **)error {

        BRGitHubApiService *api = [[BRGitHubApiService alloc] init];

        NSURL *url = [NSURL URLWithString:gitHubLogin.repositoriesPath];
        NSDictionary *headers = (gitHubLogin.repositoriesLastModified)
        ? @{@"if-modified-since": gitHubLogin.repositoriesLastModified}
        : nil;
        NSHTTPURLResponse *response = nil;
        NSMutableURLRequest *request = [api requestFor ... withHeaders:headers];
        NSData *data = [NSURLConnection sendSynchronous...Response:&response error:error];
        if (response.statusCode == 304) {
                return [NSArray array];
        }
        *lastModified = response.allHeaderFields[@"Last-Modified"];
        return json;
}
```

06_caching

BRRepositoriesService.m

# git checkout 06_caching

- ✓ Watch service in debugger
- ✓ Activity Instruments


Instruments
Record
GitHubReader
Target
Inspection Range
00:00:40
Run 1 of 1
View
Library
Instrument Detail
Filter
Instruments
Activity Monitor
VM Size
% Total Load
Net Bytes In Per Second
Net Bytes Out Per Second
Trace Highlights
Highlight By
Type of Target
% CPU
aggregated
locationd
DTMobileIS
GitHubReader
backboardd
0.00%
13.00%
26.00%
39.00%
52.00%
CPU Time
SpringBoard
CommCenter
mediaserverd
MobileMail
backboardd
00.00
4:08.248
8:16.496
12:24.744
Real Memory Usage
35.60 MB
7.17 MB
16.40 MB
31.50 MB
Dropbox
GitHubReader
SpringBoard
Weather
Real Memory Usage
Dropbox
GitHubReader
SpringBoard
Weather
New json data coming in
Cached data, no network transfer

# git checkout 07_spinners

- ✓ Percieved Performance

```objc
#import "UITableViewCell+activity.h"

- (void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:(NSIndexPath *)indexPath {

        BRGHLogin *login = (BRGHLogin *)[...
        BRRepositoriesService *service = [...

        UITableViewCell *cell = [self.tableView cellForRowAtIndexPath:indexPath];
        [cell setActivityIndicatorAccessoryView];

        [service begin... withCompletion:^(BOOL saved, NSError *error) {

                [cell clearAccessoryViewWith:UITableViewCellAccessoryDisclosureIndicator];
                [self performSegueWithIdentifier:@"SegueFromOrganizations" sender:indexPath];
        }];
}
```

07_spinners

BROrganizationsViewController.m

# Review

- Observe, Hypothesise, Experiment, Verify
- Fork the repo and try this stuff on your own
- Instruments
  - Leaks
  - Time Profiler
  - Core Data
  - Core Animations
  - Activity Monitor
- Move heavy operations off the main UI thread

# Additional Resources

- WWDC 2013
  - Core Data Performance Optimization and Debugging
  - Designing Code for Performance
- Apple Documentation
  - Threaded Core Data Example Source
    http://developer.apple.com/library/ios/samplecode/ThreadedCoreData/Introduction/Intro.html
  - Concurrency with Core Data
    http://developer.apple.com/library/ios/DOCUMENTATION/Cocoa/Conceptual/CoreData/Articles/cdConcurrency.html
  - Concurrency Programming Guide
    https://developer.apple.com/library/ios/documentation/General/Conceptual/ConcurrencyProgrammingGuide/Introduction/Introduction.html#//apple_ref/doc/uid/TP40008091
- Other Interesting Frameworks
  - github API
    http://developer.github.com/guides/getting-started/
  - AFNetworking
    https://github.com/AFNetworking/AFNetworking
  - RestKit
    https://github.com/RestKit/RestKit/


Thanks guys!
Enjoy the rest of devLink!
github.com/danielnorton
@daniel_norton