



# WOWODC '012

MONTREAL JUNE 30, JULY 1ST AND 2ND 2012

# WebObjects Optimization: EOF and Beyond

Chuck Hill, VP Development  
Global Village Consulting, Inc.

*Ranked 76th in 24th annual PROFIT 200 ranking of  
Canada's Fastest-Growing Companies by PROFIT Magazine!*

WOWODC 2012

# Session Overview

- Outline:
  - Follow the architecture
- Three kinds of optimization:
  - low effort, high effort, application specific
- Most is pretty easy

# A WOrd of Advice

- Be Productive: Measure, don't Guess
- Seek High ROI (Return On Investment)
- Premature Optimization

# Performance Measuring

- Use realistic set of data
- Beware the first request!
- jperf, jmeter, shark, range of options
- Simple as NSTimestamp and logging
- Wonder has functionality too
- WOEvent and EOEvent can be used also

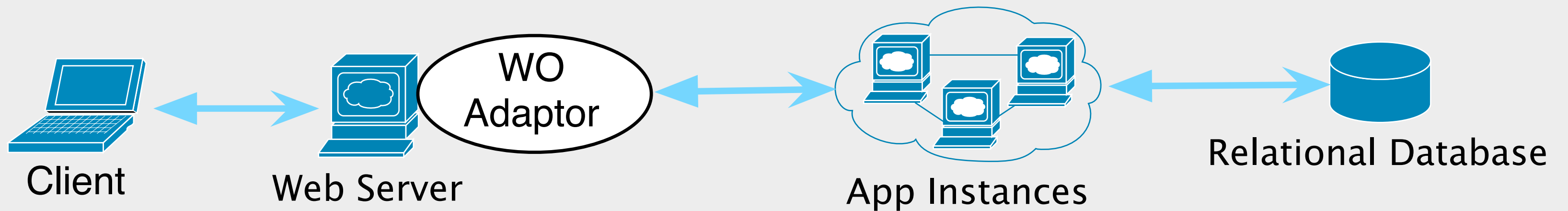


# ERProfiling and Heat Map

- From Mike Schrag and Apple
- Understand how your app is functioning
- Understand why it's slow
- Designed around WebObjects
- Page-based approach to profiling
- Look at the statistics for individual pages and actions

```
profiler: 194.71ms | <1ms:39% (1228), <10ms:24% (23), <100ms:37% (2), >=100ms:0%  
(0) | SQL: 7% (4) | D2W: 0% (0) | take:0%, invoke:0%, append:92% (all three) | all | heat off
```

# End to End



# Browser Considerations

- gzip compression
- `er.extensions.ERXApplication.responseCompressionEnabled=true`
- Minify js
- Combine CSS
- Combine images
- Minify HTML and CSS

# WebServer Side

- mod\_gzip
- mod\_deflate
- mod\_expires

```
<IfModule mod_expires.c>
```

```
ExpiresActive On
```

```
ExpiresDefault A60
```

```
ExpiresByType application/javascript A3600
```

```
ExpiresByType text/css A3600
```

```
ExpiresByType text/html A1
```



# Apache Tuning

- MinSpareServers 10
- MaxSpareServers 20
- MaxRequestsPerChild 10000
- Timeout 45
- MaxKeepAliveRequests 50000
- KeepAliveTimeout 15
- ServerLimit 2048
- ListenBackLog 511
- MaxClients 128

# WO Adaptor Settings

- FastCGI in Wonder
- Keep worker threads and listen queue size low

<b>Adaptor:</b>	<input type="text" value="WODefaultAdaptor"/> ('WODefaultAdaptor' is the default)
<b>Minimum Adaptor threads:</b>	<input type="text" value="2"/> (Applies only to the WODefaultAdaptor for WO 5.x)
<b>Maximum Adaptor threads:</b>	<input type="text" value="8"/> (Applies only to the WODefaultAdaptor for WO 5.x)
<b>Adaptor threads:</b>	<input type="text" value="8"/> (Applies only to the WODefaultAdaptor for WO 4.5.x)
<b>Listen Queue Size:</b>	<input type="text" value="4"/>

- Only default (Round Robin) load balancing works(?)
- Interleave instances across servers

# Application and Session

- `setAllowsConcurrentRequestHandling(true);`
- `setCachingEnabled(true);`
- `-WODDebuggingEnabled=false`
- `setSessionTimeOut(10 * 60);`
- `setPageCacheSize(5);`
- `setPermanentPageCacheSize(5);`

# WOCComponents

- Stateless components are harder to write but lowers memory usage
- Manual binding synchronization requires more code but less processing
- Return `context().page()` instead of null
- Lazy creation defers processing and memory usage until needed

```
public String someValue() {  
    if (someValue == null) {  
        // Create someValue here  
    }  
    return someValue;  
}
```

# Java

- ~~new Integer(8)~~ Integer.valueOf(8)
- ~~StringBuffer~~ StringBuilder
- Null references when not needed
- Heap Size
  - Xms256m -Xmx512m
- Google for advanced heap size tuning articles

# Using the Snapshot Cache

- Rows for fetches objects stored in EODatabase as snapshots
- Snapshots have a Global ID, retain count, and fetch timestamp
- Using the row snapshots is *fast*
  - following relationships
  - `objectForGlobalID`, `faultForGlobalID`
- Consider object freshness needs
- Fetch Specs go to database
- Raw rows go to database



# EOSharedEditingContext

# ERXEnterpriseObjectCache

- Both address “read mostly” frequent access data
- Both prevent snapshots from being discarded
- EOSharedEditingContext requires few changes
- ... but may introduce bugs. Maybe.
- ERXEnterpriseObjectCache requires more work
- Key based object access (or global ID)
- ... but you have the source and it is commonly used
- EOModel “Cache in Memory” never refreshes

# ERXEnterpriseObjectCache Usage

```
ERXEnterpriseObjectCache cache = new ERXEnterpriseObjectCache(  
    entityName, keyPath, restrictingQualifier, timeout,  
    shouldRetainObjects, shouldFetchInitialValues,  
    shouldReturnUnsavedObjects);
```

```
ERXEnterpriseObjectCache cache = new ERXEnterpriseObjectCache(  
    "BranchOffice", branchCode, null, 0, true, true, true);
```

```
public static BranchOffice branchWithCode(E0EditingContext ec, Long id){  
    return (BranchOffice)officeCache().objectForKey(ec, id);  
}
```

```
BranchOffice montrealBranch = BranchOffice.branchWithCode(ec, "MTL");
```

# Mass Updates

- Sometimes EOF is not the best solution
- e.g. bulk deletions will fetch all of the EOs first
- ERXEOAccessUtilities
  - deleteRowsDescribedByQualifier()
  - updateRowsDescribedByQualifier()
  - insertRows()
- ERXEOAccessUtilities.evaluateSQLWithEntityNamed

# Using Custom SQL

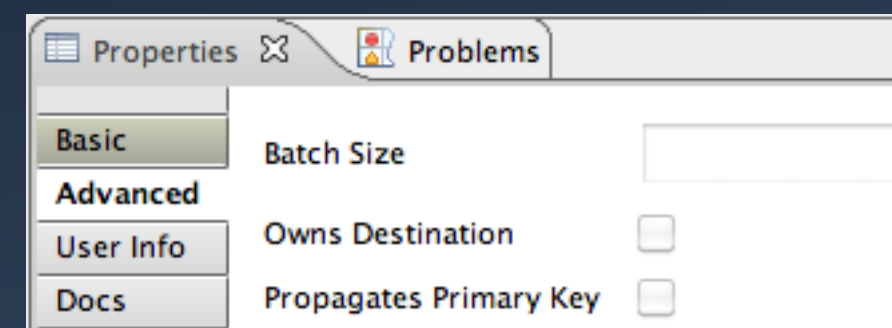
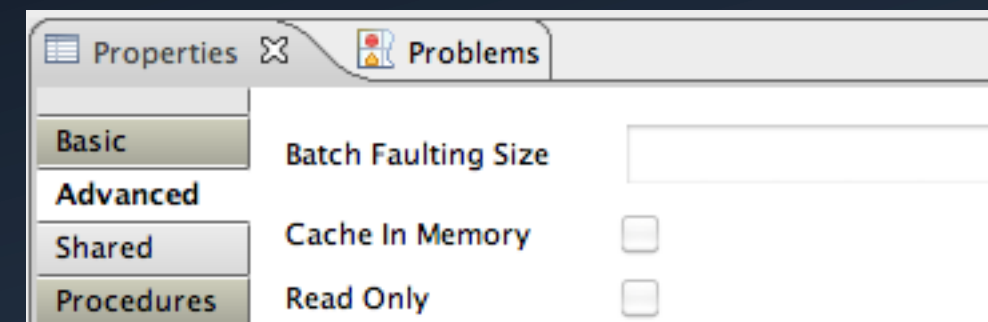
- Sometimes there is no other way
- Easier than writing a custom EOQualifier
- EOUtilities
- ERXEOAccessUtilities

# ERXBatchingDisplayGroup

- Drop-in replacement for WODisplayGroup
- Alternative to limiting data set size
- Fetches one batch of EOs at a time
- Low memory and fetch overhead
- Still fetches all Primary Keys
- Kieran's LIMIT option
- ERXBatchNavigationBar
- AjaxGrid and AjaxGridNavBar

# Batch Faulting (Fetching)

- Optimistically faults in objects
- Set in EOModel
- Entity or Relationship
- How big should a batch be?
- Two is twice as good as none
- 10 - 20 is a good guess





# Prefetch Relationships

- Extension/alternative to batch faulting
- Fetches everything at once
- Allow for more precise tuning than Batch Faulting
- Only useful if you need all / most of the objects
- `EOFetchSpecification.setPrefetchingRelationshipKeyPaths()`
- Can only follow class property relationship from root
- One fetch per relationship key path with migrated qualifier
- Not optimal if most of objects are in snapshot cache

# ERXBatchFetchUtilities

- Alternative to pre-fetching and batch faulting
- Very focused batching of fetches
- Efficiently batch fetch arbitrarily deep key paths
- `batchFetch(NSArray sourceObjects, NSArray keypaths)`
- One Gazillion options to control fetch

# When to use Raw Rows

- Data ONLY, no logic, no methods, no code, no anything
- NSDictionary of key/value pairs
- Use with a lot of data from which you only need a few EOs
- EOFetchSpecification, EOUtilities, ERXEOAccessUtilities
- Late promotion with:
  - `EOUtilities.objectFromRawRow(ec, entityName, row)`
  - `ERXEOControlUtilities.faultsForRawRowsFromEntity(ec, rows, entityName)`


# EOFetchSpecification Limit

- `setFetchLimit(int limit)`
- This may not do what you expect
- The standard is to fetch ALL rows and limit in memory
- `prefetchingRelationshipKeyPaths` do not respect LIMIT
- Check the SQL!
- Wonder fixes SOME databases to LIMIT at database
- YOU can fix the rest! Contribute to Wonder!
- `ERXEOControlUtilities.objectsInRange(ec, spec, start, end, raw)`

# Don't Model It! Just Say NO!

- Avoid unnecessary relationships
- Can Model It != Should Model It
- Relationships from look-up tables to data
- Address TO Country ✓ Country TO Address ✗
- EOF will fault in ALL of the data rows, VERY slow
- Do. Not. Do. This.
- Fetch the data IF you ever need it

# Factor out large CLOBs

- Simple and easy to avoid
- Fetching large CLOBs (or BLOBs) consumes resources
- Move LOB values to their own EO
- CLOB EO is to-one and Owns destination
- `object.clobValue()`  `object.clob().value()`
- large values are fetched only on demand



# Model Optimization

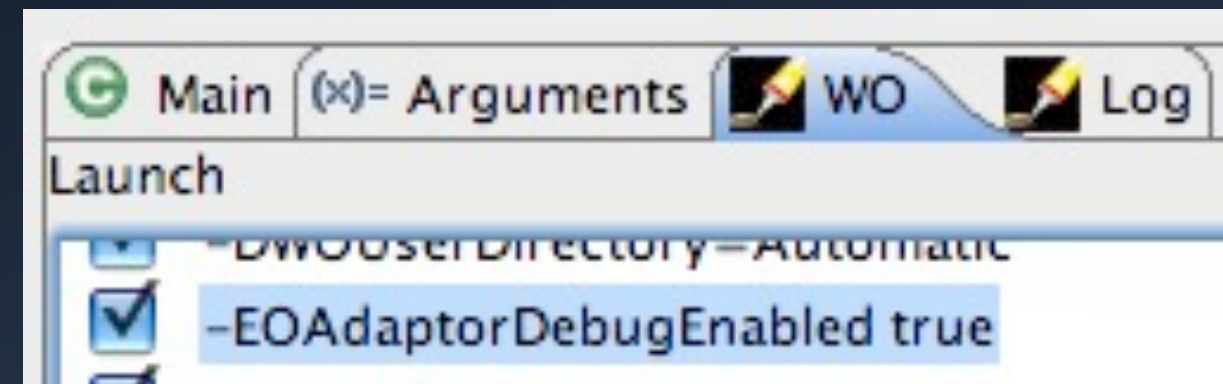
- Trim the fat
- Map multiple Entities to same table (read-only, careful!)
- Reduce number of attributes locked
- De-normalize (views, flattening)
- Keep complex data structures in LOBS
- Stored Procedures

# Inheritance and Optimization

- Inheritance can be very useful
- Inheritance can be very slow
- Keep hierarchies flat
- Avoid concrete super classes
- Single Table inheritance is the most efficient
- Vertical inheritance is the least efficient

# Monitor the SQL

- easiest, cheapest, highest payback performance tuning
- `-EOAdaptorDebugEnabled true`
- Watch for:
  - repeated single row selects
  - slow queries (more data makes more obvious)
- Check for:
  - indexes for common query terms



# ERXAdaptorChannelDelegate

## SQLLoggingAdaptorChannelDelegate

- Tracks and logs the SQL that gets sent to the database
- ERXAdaptorChannelDelegate
  - thresholds for logging levels
  - filter by Entity (regex)
- SQLLoggingAdaptorChannelDelegate
  - CSV formatted log message output for Excel analysis
  - can log data fetched
  - can log stack trace of fetch origin

# ERChangeNotificationJMS

- Synchronizes EOs and snapshots between application instances
- Can reduce fetching
- Can reduce need to fetch fresh data
- Will reduce save conflicts

# Join Table Indexes

- Join tables only get one index
- Some EOF generated SQL can't be optimized
- Results in table scan
- Manually add complementary index



# Database Tuning

- Check the plan, Stan
- Cache, cache, cache
- RAM, RAM, RAM
- Check hit ratio and tune cache size

# SURVs Optimization

**SURVS**   
*asking for you*

# Counters and Concurrency

- **Situation:** you need to count records according to some criteria
- **Problems:**
  - counting with a query is too slow
  - so, create a counters row and update it in real time for new/updated data
  - thousands of users creating thousands of events on a short time
  - huge resource contention for the counter, lots of retries

# Solution

- **Solution:** create several sub-counters!

ID	Counter Identifier	SubCounter Number	Value
	Apples	0	567
	Apples	1	345
	Apples	2	487
	Oranges	0	1435
	Oranges	1	1372
	Strawberries	0	45
	Cherries	0	390

- Counter identifier is one or more columns with whatever you need to identify your counter (FK to other tables, whatever).
- SubCounter Number is a number identifying one sub counter for the counter identifier

# How Does it Work?

- Define a maximum number of sub counters
- When reading, simply select all counters for your identifier, and obtain the sum of the value column.
- To update, SubCounter is random number 0 ... max counters - 1
- That's it. You just reduced the probability of having OL failure and repeating by a factor of 10

ID	Counter Identifier	SubCounter Number	Value
	Apples	0	567
	Apples	1	345
	Apples	2	487
	Oranges	0	1435
	Oranges	1	1372
	Strawberries	0	45
	Cherries	0	390





# WOWODC '012

MONTREAL JUNE 30, JULY 1ST AND 2ND 2012



## Q&A

WebObjects Optimization: EOF and Beyond

Chuck Hill

Global Village Consulting