David Kuster
dave@talldave.net
@ofTallDaveFame

Hibernate Metrics

Paying Attention to the ORM
Behind the Curtain

About Me

- Grails dev ~4 years
- Java/JEE dev ~10 years prior
- Bootstrapped startups to Fortune 500
- Developer of Grails Command Center
- “You're really good at writing your codes.”
  - my wife

Agenda

- The landscape
- The solutions
- The problems that the solutions have uncovered
- And some fixes

A Caveat

- Hibernate is a big topic
- I came to Hibernate through GORM
- There are definitely gaps in my knowledge on this topic

What GORM Promises

The Reality

And…

“Object-relational mapping is the Vietnam of Computer Science.”

- Ted Neward

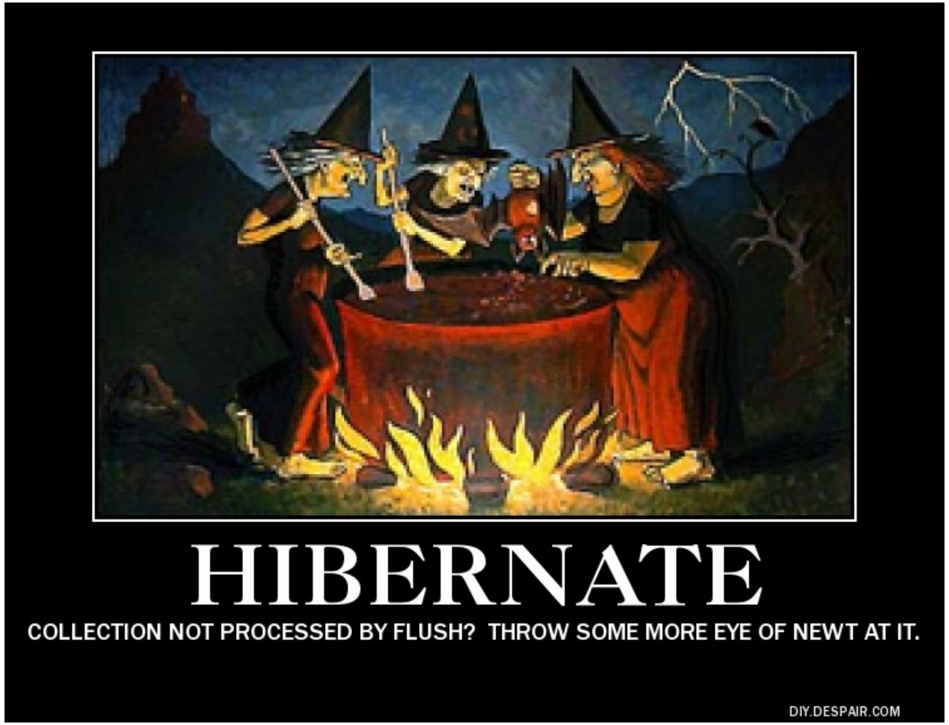
I love the smell of object-relational impedance mismatch in the morning

Best vs Worst Case Scenarios

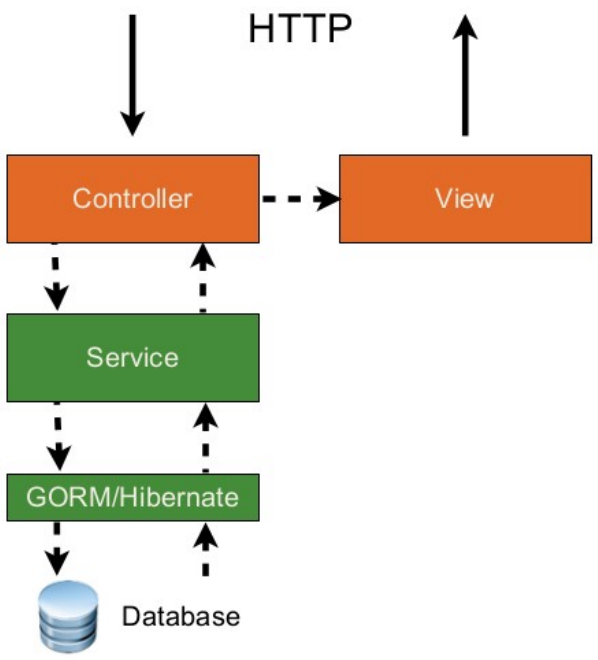
The Point Being

PAY ATTENTION TO THE DATABASE

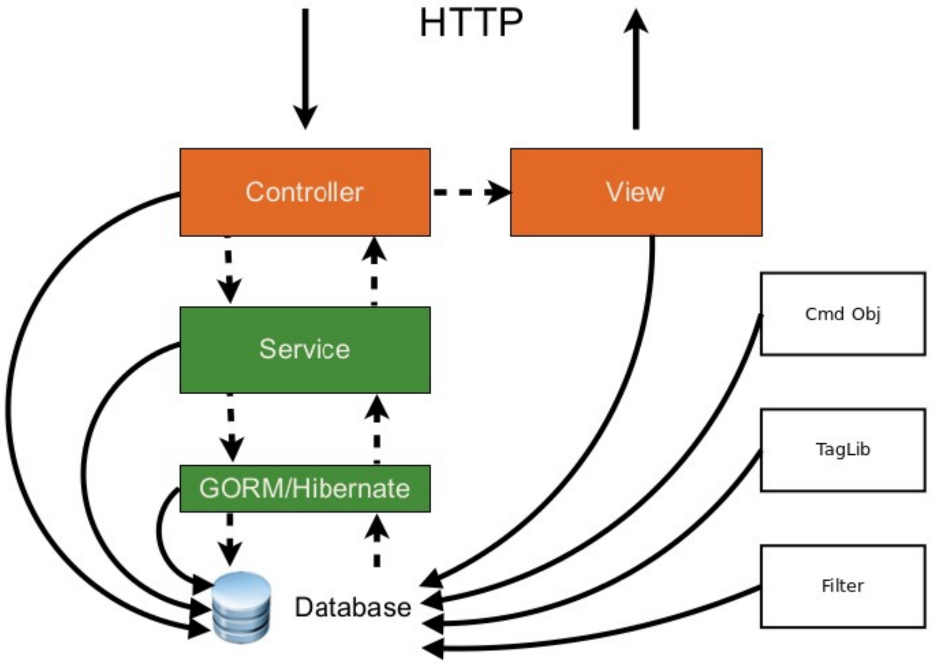
Any Sufficiently Advanced Technology Is Indistinguishable From Magic


HIBERNATE
COLLECTION NOT PROCESSED BY FLUSH? THROW SOME MORE EYE OF NEWT AT IT.
DIY.DESPAIR.COM

Conceptual Model



Database Access From Anywhere



Agenda

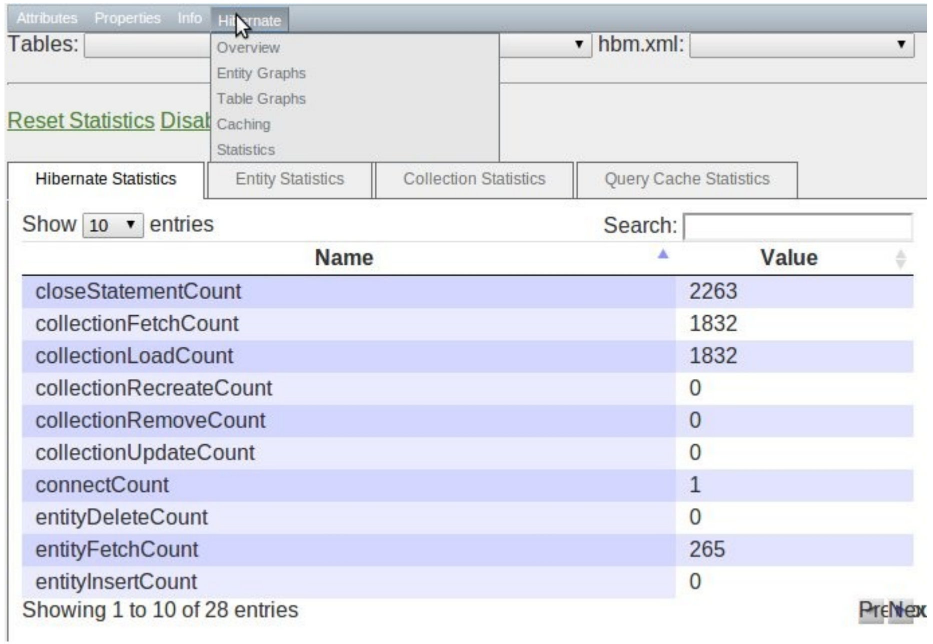
- The landscape
- The solutions
- The problems that the solutions have uncovered
- And some fixes

A Well-Trod Path

- Many have come this way before
- A number of plugins (and external/paid tools) that allow us to pay attention to the DB

Hibernate App-Info

- Extension to App-Info Plugin
- Add mixins to Config.groovy
- /appName/adminManage
- A ton of info, but not easily browsable

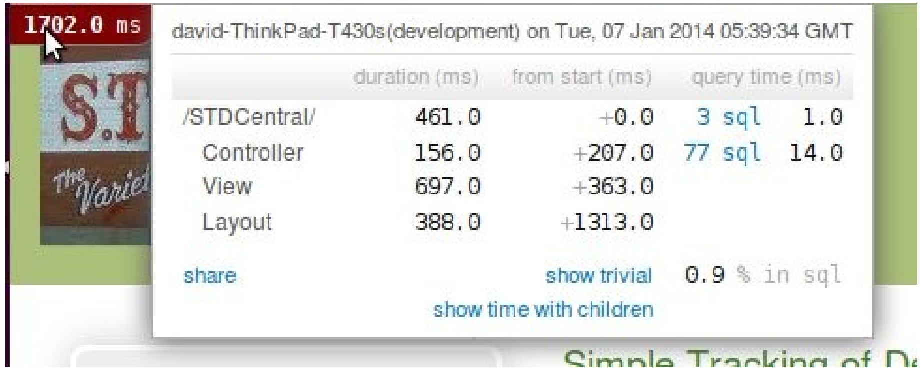


Name	Value
closeStatementCount	2263
collectionFetchCount	1832
collectionLoadCount	1832
collectionRecreateCount	0
collectionRemoveCount	0
collectionUpdateCount	0
connectCount	1
entityDeleteCount	0
entityFetchCount	265
entityInsertCount	0

Showing 1 to 10 of 28 entries

Mini-Profiler Plugin

- Built on top of Profiler Plugin
- Mimics StackExchange MiniProfiler (.NET and Ruby)
- Add <miniprofiler:javascript/> to main.gsp
- Heads up display
- Timing and SQL statements per artefact type



1702.0 ms

david-ThinkPad-T430s(development) on Tue, 07 Jan 2014 05:39:34 GMT

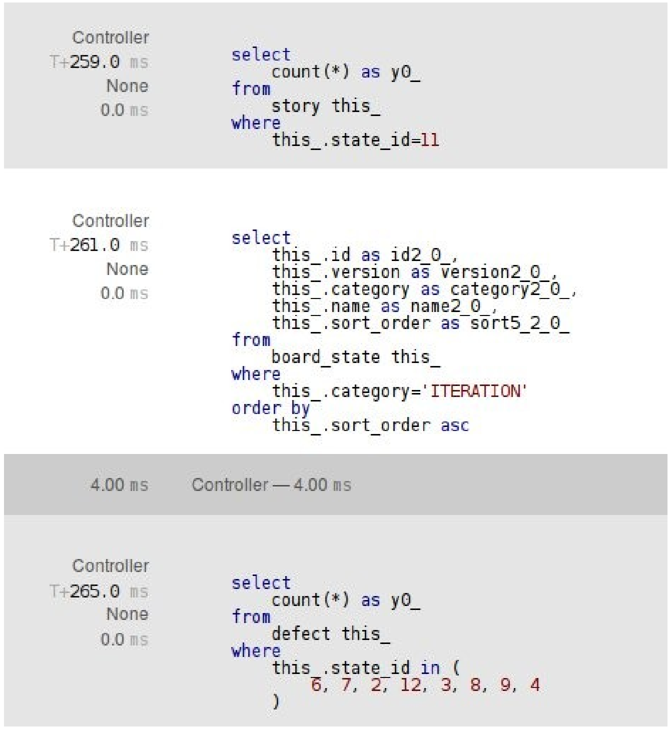
	duration (ms)	from start (ms)	query time (ms)	
/STDCentral/	461.0	+0.0	3 sql	1.0
Controller	156.0	+207.0	77 sql	14.0
View	697.0	+363.0		
Layout	388.0	+1313.0		

share   show trivial   0.9 % in sql

show time with children

Mini-Profiler Plugin

- View SQL queries with actual params
  - great debugging tool
- No summary of repeated queries
- JS dialog a little flaky
- Still, recommended

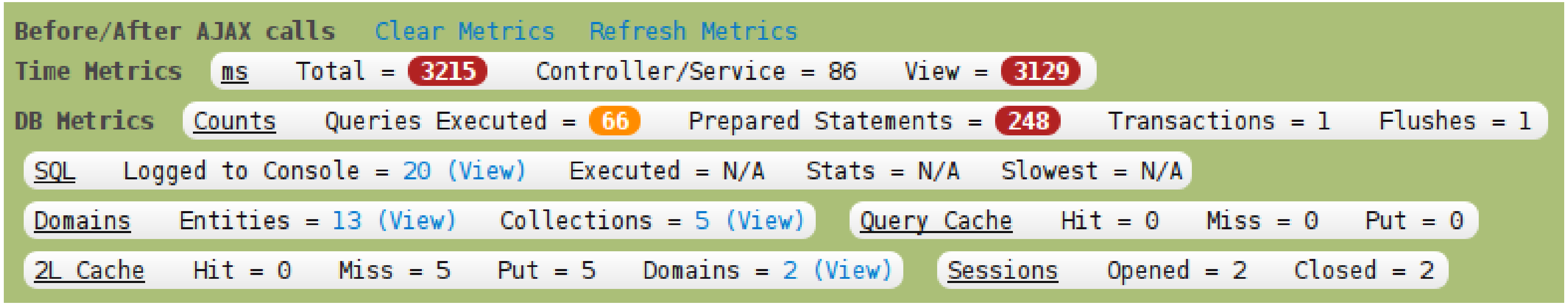

```
    Controller
T+259.0 ms          select
      None              count(*) as y0_
    0.0 ms          from
                        story this_
                    where
                        this_.state_id=11

    Controller
T+261.0 ms          select
      None              this_.id as id2_0_,
    0.0 ms              this_.version as version2_0_,
                        this_.category as category2_0_,
                        this_.name as name2_0_,
                        this_.sort_order as sort5_2_0_
                    from
                        board_state this_
                    where
                        this_.category='ITERATION'
                    order by
                        this_.sort_order asc

   4.00 ms      Controller — 4.00 ms

    Controller
T+265.0 ms          select
      None              count(*) as y0_
    0.0 ms          from
                        defect this_
                    where
                        this_.state_id in (
                            6, 7, 2, 12, 3, 8, 9, 4
                        )
```

Hibernate Metrics

- Not yet released
- Focus is on Hibernate behavior and domain objects as well as SQL generated/executed
- Programmatically enables Statistics API, doesn't wrap DB driver
- Thus, able to turn on/off at will
- Heads up display ala Mini Profiler Plugin

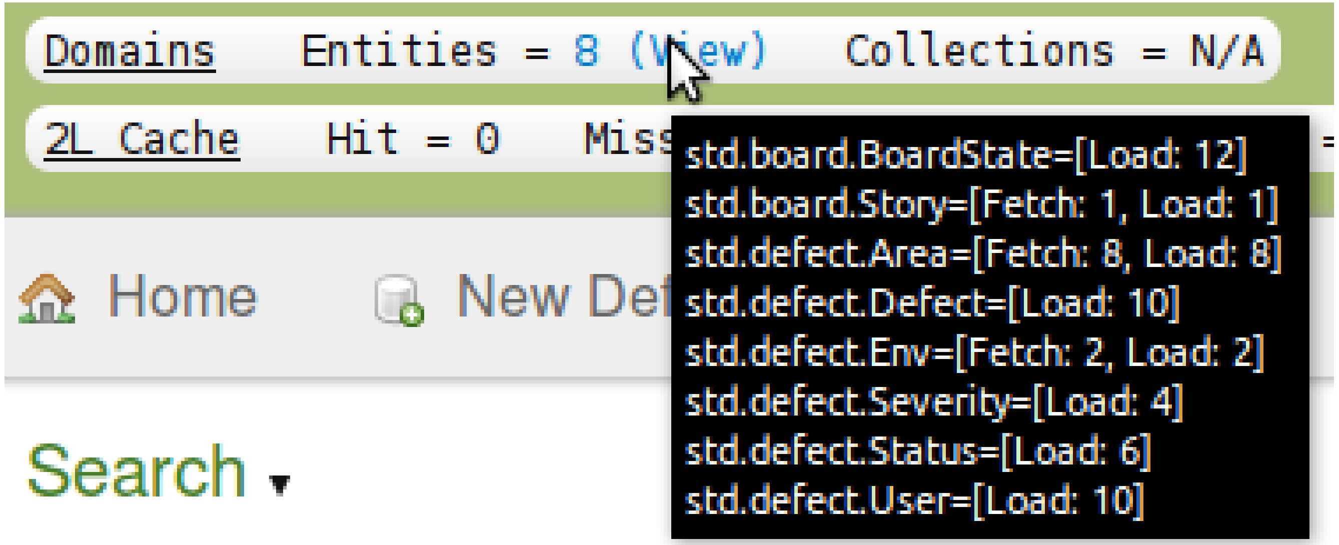
Hibernate Metrics

- Timing and database/Hibernate info
- Intercepts logSql console output, groups queries by execution count
- Hibernate Stats doesn't report on criteria queries (until version 4.3+)



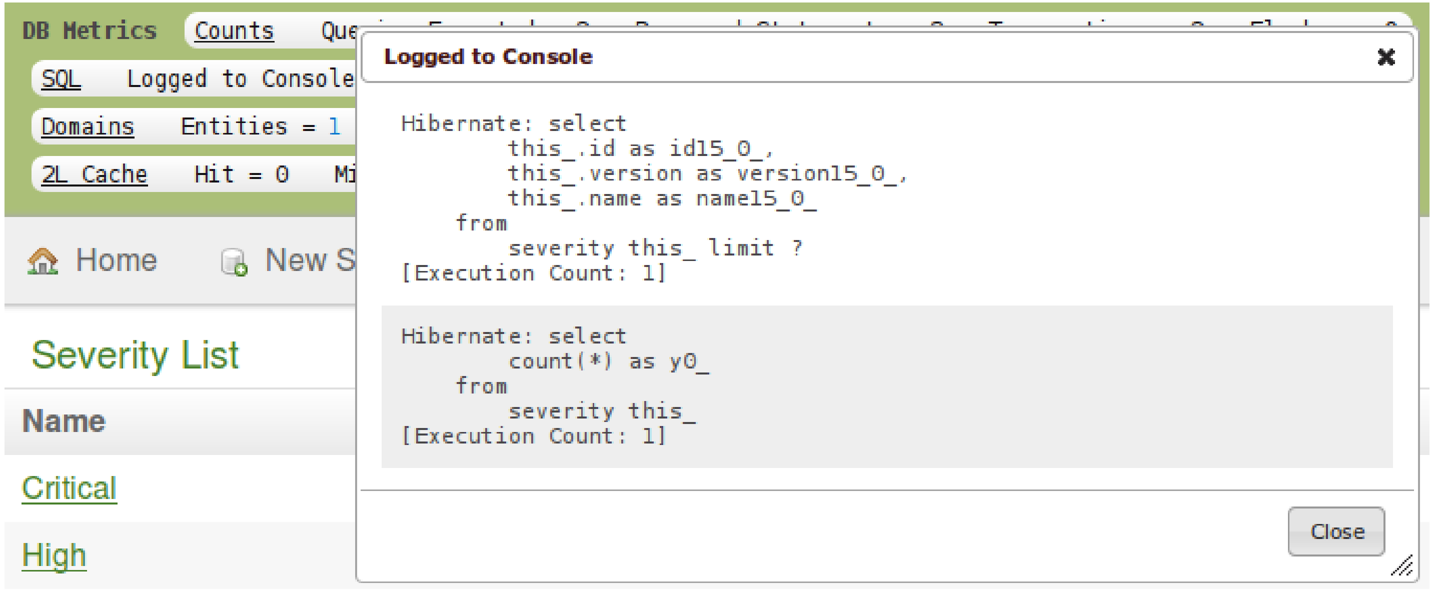
```
Before/After AJAX calls   Clear Metrics   Refresh Metrics
Time Metrics  ms   Total = 3215   Controller/Service = 86   View = 3129
DB Metrics  Counts   Queries Executed = 66   Prepared Statements = 248   Transactions = 1   Flushes = 1
 SQL   Logged to Console = 20 (View)   Executed = N/A   Stats = N/A   Slowest = N/A
 Domains   Entities = 13 (View)   Collections = 5 (View)   Query Cache   Hit = 0   Miss = 0   Put = 0
 2L Cache   Hit = 0   Miss = 5   Put = 5   Domains = 2 (View)   Sessions   Opened = 2   Closed = 2
```

Hibernate Metrics



```
std.board.BoardState=[Load: 12]
std.board.Story=[Fetch: 1, Load: 1]
std.defect.Area=[Fetch: 8, Load: 8]
std.defect.Defect=[Load: 10]
std.defect.Env=[Fetch: 2, Load: 2]
std.defect.Severity=[Load: 4]
std.defect.Status=[Load: 6]
std.defect.User=[Load: 10]
```

Hibernate Metrics



```
Hibernate: select
        this_.id as id15_0_,
        this_.version as version15_0_,
        this_.name as name15_0_
    from
        severity this_ limit ?
[Execution Count: 1]
```

```
Hibernate: select
        count(*) as y0_
    from
        severity this_
[Execution Count: 1]
```

Hibernate Metrics – Integration

- Not actually released yet, so…
- Outside plugins block of BuildConfig

```
// BuildConfig.groovy

grails.plugin.location.'hibernate-metrics' =
"/path/to/cloned/github/repo"
```

Hibernate Metrics – Integration

- Add display option to layout
- Turn on in dev via link
- Turn on in prod via URL

```
<!-- show normal header if not enabled -->
<perfMetrics:isNotEnabled>
    <a href="/"><img src="logo.jpg"/></a>
</perfMetrics:isNotEnabled>

<!-- otherwise hide header, show stats -->
<perfMetrics:isEnabled>
    <perfMetrics:metrics />
</perfMetrics:isEnabled>

<!-- enable/disable links in dev env -->
<perfMetrics:devEnvControl />

// enable/disable directly (prod)
/myApp/hibernateMetrics/enable
/myApp/hibernateMetrics/disable
```

Hibernate Metrics – Integration

- Programmatic integration possible
- Primarily used from the Console Plugin

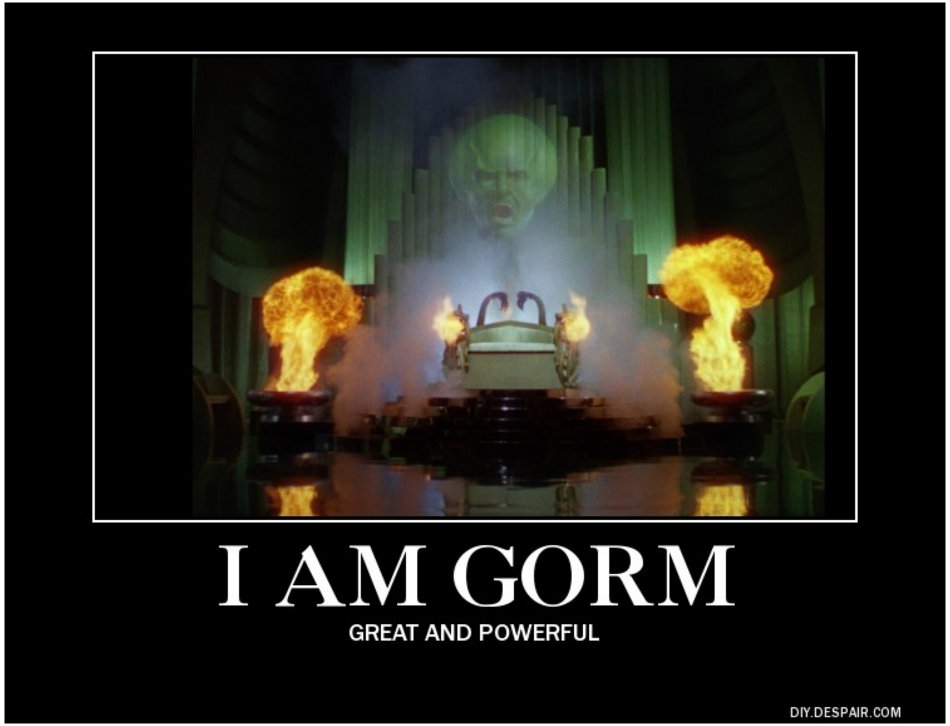
```
HibernateMetrics.withSqlLogging {
  // execute code that you want SQL to be logged for
}

// println output
Time Metrics:
  Total Time (ms) = 30
  Controller/Service (ms) = 28
  View (ms) = 2
DB Metrics:
  Total Queries = 2
  Prepared Statements = 2
  Logged SQL = ...
  Entity Info = std.board.BoardState = [Load: 12]
  ...
  Sessions Opened = 0
  Sessions Closed = 0
  Transaction Count = 0
  Flush Count = 0
```

Demo

Agenda

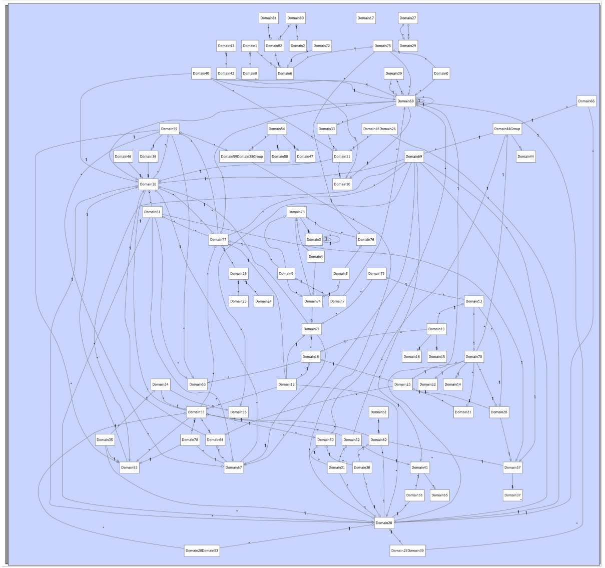
- The landscape
- The solutions
- The problems that the solutions have uncovered
- And some fixes

Some of the Things I've Learned



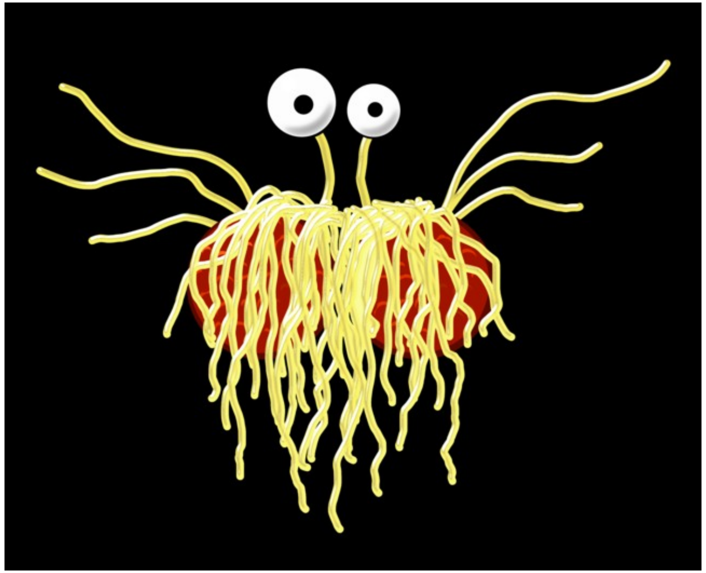
Environments

80+ domains



Environments

190+ domains



Queries in a Loop

- Easy win to minimize number of round trips to the DB

```
// results in query per id
params.inputIds.each {
  list << Defect.get(it)
}

// use .getAll() instead - single query
list << Defect.getAll(params['inputIds'])
```

Queries in a Loop - Continued

- Pay attention any time you're looping
- Queries may be wrapped/obfuscated by other logic

```
// this logic works, but can we be smarter
about it?
params.inputIds.each { id ->
  story.addToDefects(
    buildDefectWithSpecialLogic(
      Defect.get(id), user, client))
}

// single query instead
List d = Defect.getAll(params.inputIds)
d?.each { defect ->
  story.addToDefects(
    buildDefectWithSpecialLogic(
      defect, user, client))
  }
}
```

Navigating Obj Structure To Get Minimal Data

- A lot of data is a terrible thing to waste

```
// loads full Story, Status, Comment objects
Story.findAllByStatus(
  Status.findByName("Completed")
)*.comments*.user?.name


// loads just the data needed, in one query
Story.createCriteria.list() {
  comments {
    user {
      property('name')
    }
  }
  status {
    eq('name', 'Completed')
  }
}
```

Navigating Obj Structure - Continued

- Happens in other scenarios too

```
// a similar thing
story.subTasks*.estimates.find {
  it.id == id
}


// loads just the data needed, in one query
Story.createCriteria.get() {
  subTasks {
    estimates {
      eq('id', id)
    }
  }
}
```

hasOne Always Eager Fetches

- If the OtherDomain in the hasOne is expensive to load this can be an unwelcome surprise

```
class MyDomain {
  static hasOne = [alwaysAlongForTheRide:
OtherDomain]
}


class OtherDomain {
  static hasOne = [a:A, b:B, c:C, d:D, ...]
  static mapping = {
    collection1 fetch:'join'
    collection2 fetch:'join'
    collection3 fetch:'join'
    collection4 fetch:'join'
    collection5 fetch:'join'
    ...
  }
}
```

N+1 Queries Problem

- Default per relationship is lazy loading

```
class Story {
  static hasMany = [defects:Defect]
}

// iterating over collection causes N+1
Story.get(id)?.defects?.each { defect →
  hoursCount += defect.hoursSpent
}
```

Fixing N+1 Queries

- Fetch:'join' – join table & get all data in a single query
- Lazy:false – immediately execute second query to load collection
- Override per query

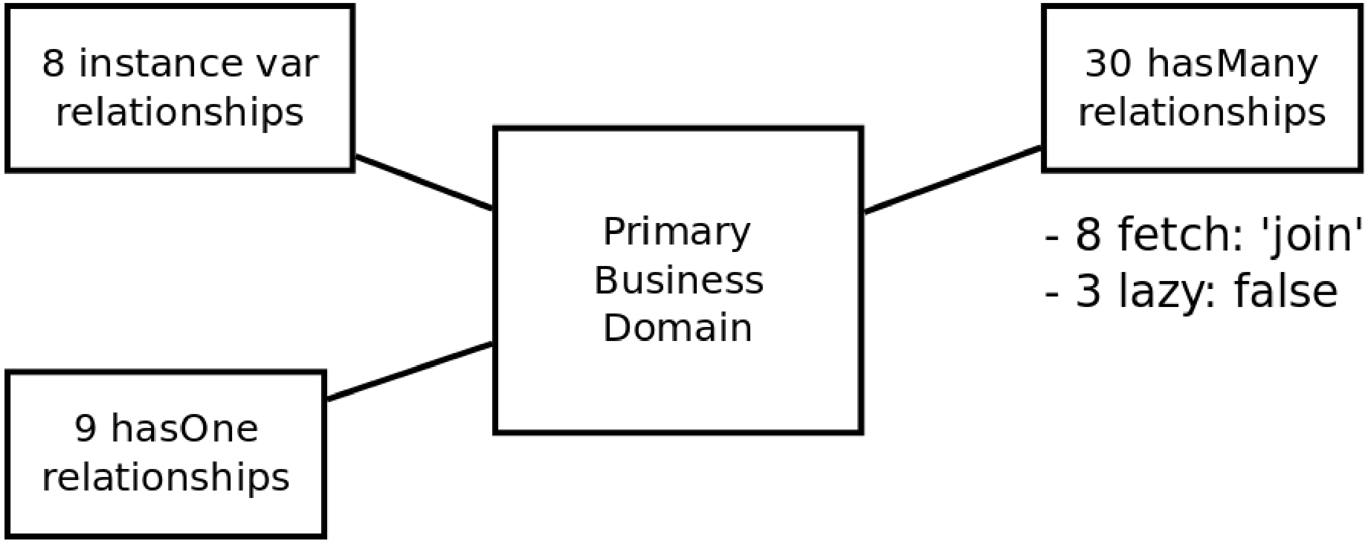
```
class Story {
  static hasMany = [defects:Defect]

  // pick one
  static mapping = {
    //defects lazy:false
    //defects fetch:'join'
  }
}

// specify at query time
Story.list( [fetch:[defects:'join']] )
```

Another N+1 Gotcha (?)

- Not naming belongsTo seems to guarantee N+1 queries

```
class Story {
  static hasMany = [defects:Defect]
}

// creates a join table
class Defect {
  static belongsTo = [Story]
}

// no join table, story_id on Defect table
class Defect {
  static belongsTo = [story:Story]
}
```

Real World Example



- 8 fetch: 'join'
- 3 lazy: false

Custom Validators

- Validators can get called a lot…
- At the very least use save(validate:false) if manually checking validate()

```
class Story {
  static constraints = {
    name validator: { val, obj ->
      doQueryToCheckUniqueness(val, obj)
    }
  }
}



// save() will call validate by default
if (story.validate()) {
  // so don't call it twice
  story.save(validate:false)
}
```

Where Queries Don't Use 2nd Level Cache

- Where queries create DetachedCriteriaObject
- Not associated to a Hibernate Session
- Thus, no second-level cache

```
class MyConfig {
  static mapping = {
    cache true // use 2nd level cache
  }
}

// nice syntax, always going to hit the DB
def configVal = MyConfig.where {
  name == configName
  customer == currentCustomer
  user == user
}.get()
```

Another 2nd Level Cache Issue

- A misplaced colon makes a big difference

```
class MyConfig {
  static mapping = {
    cache: true // fail
  }
}
```

Loading Data, Ignoring Data

- View gets refactored
- Controller or service is still retrieving data

```
// controller action
def show() {
  [users:User.list(),
    severities:Severity.list(),
    environments:Env.list()]
}

// view
<g:each var="user" in="${users}">
  ..
</g:each>

<g:each var="severity" in="${severities}">
  ..
</g:each>
```

Collections load all instances when adding

- hasMany are Sets by default
- Set guarantees uniqueness
- Really only comes into play with huge collections

When Groovy SQL Is Better

- Batch jobs
  - especially big, interrelated data
- Real world example
  - converting from domains, dynamic finders, etc to Groovy SQL
  - job went from 1 week+ to 2-4 hours
- Have to do some things manually
- Pay attention to the downsides

```
// possible SQL injection sql.execute('select * from x where y = ' + input)

// avoids it
sql.execute('select * from x where y = ? and z = ?', [inputA, inputB])

// pay attention to String vs GString
sql.eachRow(“select * from x where y = ${y}”)
// won't work

sql.eachRow(“select * from x where y = ${y}” as String)
// will work
```

Fix It, Yo

- Don't prematurely optimize
- Proper DB indexes for your queries
- Warm it up – or cool it down
- A lot of it is “it depends”

Hibernate Metrics – The Future

- Data is not tracked over time
- Charts, graphs
- Integration with Integration Tests
- Actually release the damn thing

Thank You For Not Sleeping

(I hope)

Questions?

Links / Credits

- Hibernate Metrics Plugin
  https://github.com/davidkuster/hibernate-metrics
- Ted Neward – The Vietnam of Computer Science
  http://blogs.tedneward.com/2006/06/26/The+Vietnam+Of+Computer+Science.aspx
- Burt Beckwith – Advanced GORM – Performance, Customization and Monitoring
  http://infoq.com/presentations/GORM-performance
- StackExchange MiniProfiler Plugin
  http://miniprofiler.com
- Tom Dunstan – Debugging Grails Database Performance
  http://skillsmatter.com/skillscasts/3785-debugging-grails-database-performance
- Slides
  http://talldave.net

Links / Credits

- The Wonderful Wizard of Oz
  http://www.read.gov/books/oz.html
- The Wizard of Oz
  http://thewizardofoz.warnerbros.com
- Flying Spaghetti Monster
  http://dextermurphy.deviantart.com/art/Flying-Spaghetti-Monster-244993860