Pursuing Performance in Store

Algorithms, Queries and Schemas

By Tom Robinson

SIMPLIFICATION THROUGH INNOVATION®

Welcome
September 11, 2013

What is Store ?

- Store is a Version Control System for Cincom Smalltalk™
- Uses a Relational Database as the repository
- Versions
  - Bundles and Packages (components)
  - Namespaces
  - Class Definitions
  - Class Extensions (methods only)
  - Shared Variables
  - Methods

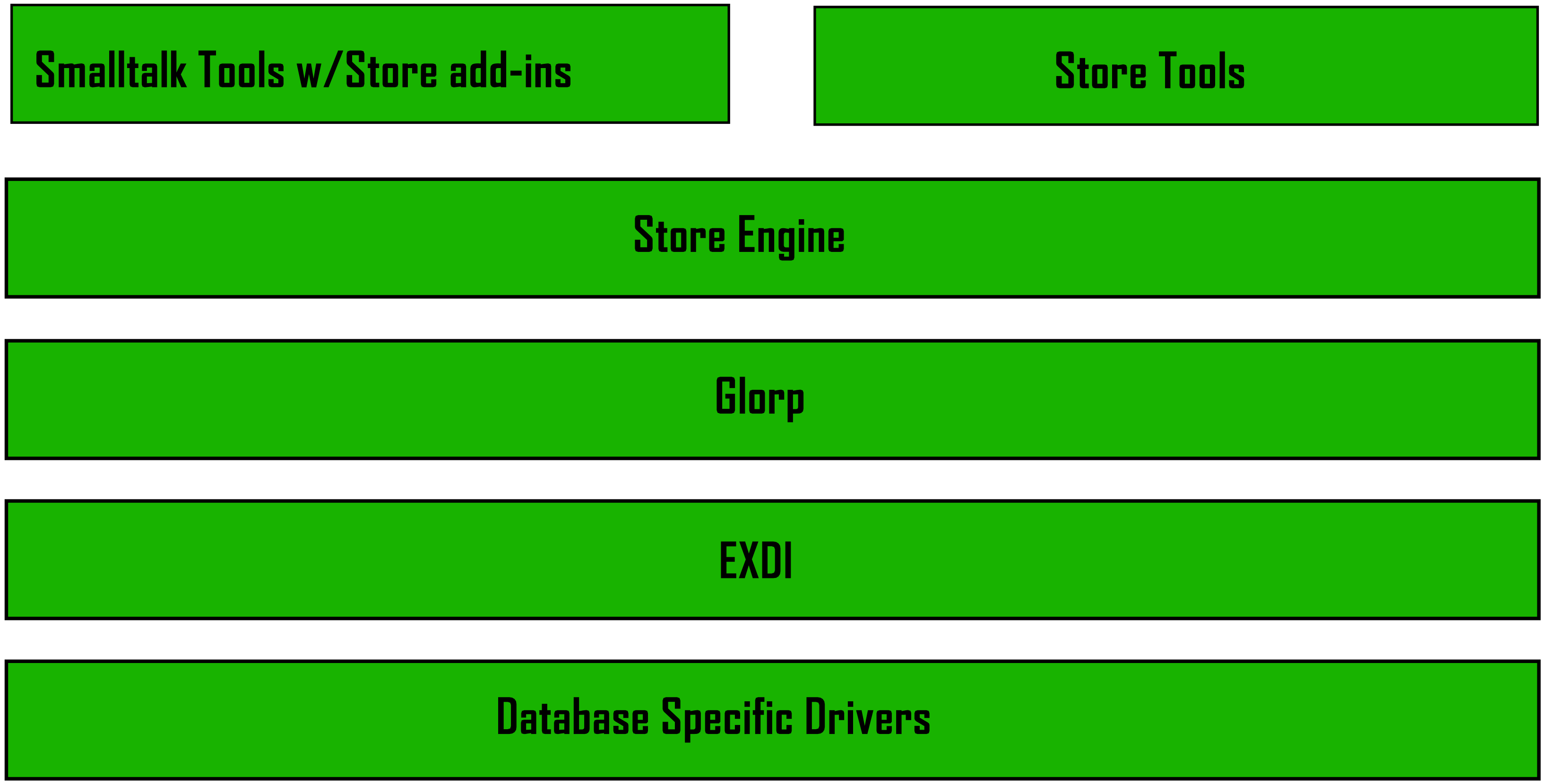
Store Capabilities

- Provides Tools for:
    - Publishing Bundles/Packages
    - Merging Bundle/Package versions into the image
    - Loading all version-able entities
    - Listing, Browsing, Comparing all version-able entities
    - Administration
- Supports object queries using Glorp

Isn’t A Relational SCCS Crazy?

- It might be….for languages that save source code in text files. Would you like to compare whole files or class and method versions?
- Relational is actually a good fit
  - 25 tables + their indexes – this is not like mortgages, insurance policies or shipping containers
  - Most enterprises
    - already use an RDB
    - already employ DBAs
    - already have other database management tools

Store Structure



Why Isn’t Store Fast Already?

It Is….

- Cincom engineers don’t find Store to be an impediment to their work
- Most customers find Store “fast enough”

…but…

Store Needs To Be (And Can Be) Faster

- Everyone likes ***faster***
- Original schema issues need fixing
- Move to Glorp (to allow addressing schema issues) focused on correctness
- Customers use Store differently than Cincom does
  - Large codebases, much larger packages and bundles
  - Extensive use of Store to version files
- Cincom needs larger codebases to benchmark against

Algorithms, Queries and Schemas

- Algorithms – making Store run faster in the image
  - More intelligent use of sessions and caching
  - Minimizing the effects of larger codebases
- Queries – making smarter database requests
  - Asking for the right amount of data
  - At the right time
- Schema
  - Needs to match the data and usage patterns
  - Shouldn’t impose unintentional constraints on usage

Making Store Run Faster in the Image

ALGORITHMS

Improving #reconcileMethods

```
StorePackage>>#reconcileMethods

    (self methods isEmpty and: [self previous methods isEmpty])
            ifTrue: [^self].

    methods size = self previous methods size
            ifFalse: [self markModified].

    methods do: [:each |
            each definition: (   self reconcileMethod: each definition)].
```

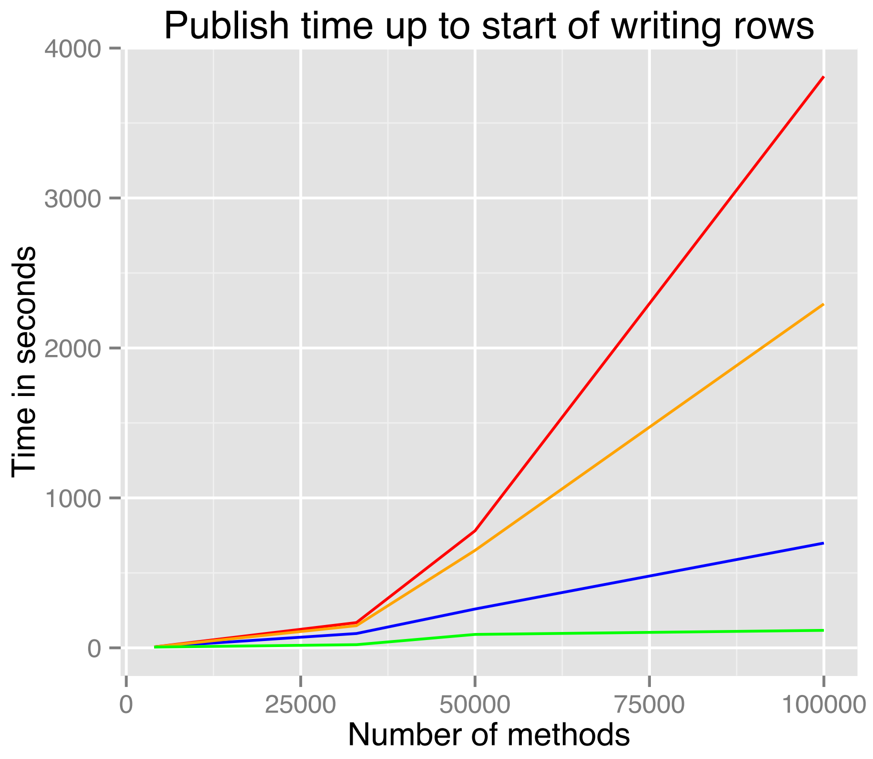
Inside #reconcileMethods

```
#reconcileMethods
    #reconcileMethod: aStoreMethod
        #basicReconcileMethod: aStoreMethod
            | exactMatch |
            self previous isNil ifTrue: [^nil].
            exactMatch := self previous methods
                detect: [:each | aStoreMethod reconcilesWith: each definition]
                ifNone: [nil].
            ^exactMatch notNil
                ifTrue: [exactMatch definition]
                ifFalse: [nil].
```

Time To Prepare To Write Rows

Package	Methods	Total Time (sec)	Percent reconcile Methods	Time for reconcile Methods
Internationalization	4,208	6.4	59.3	3.8
Jun	33,794	168.7	66.5	111.7
PerfTestPkg1	50,000	780.9	77.6	605.3
PerfTestPkg2	100,000	3811.2	81.4	3102.1

Changes In Time To Prepare To Write Rows



Revised #reconcileMethods (VW7.10.1)

- Iterates once over the collection of methods from the previous version, building a dictionary with selector and className as the key and the method as the value.
- Iterates once over the collection of methods in the current version, does a dictionary lookup to find the previous version if it exists and does a single comparison.
- **Even with 100,000 methods and 500 classes, this change doesn't save time when comparing class definitions, namespaces or shared variables.**

Making Smarter Database Requests

QUERIES

Query 1: Method Versions

- Old version retrieved one StoreMethodInPackage per package version (and frequently many per method version) and discarded duplicates
- New version retrieves the StoreMethodInPackage with the earliest timestamp for each method version, so no duplicates

Old Method Versions Query

```
allVersionsWithName: aString inClass: aClassName in: aSession
    | query session objects |
    session := aSession ifNil: [StoreLoginFactory currentStoreSession].
    query := Query
            read: self
            where: [:eachMethod | eachMethod definition name = aString
                    AND: [eachMethod definition className = aClassName]].
    query alsoFetch: #definition.
    …(additional alsoFetch:  statements removed)…
    query alsoFetch: [:each | each package].
    query orderBy: [:each | each definition timestamp descending].
    query orderBy: [:each | each package timestamp].
    objects := session execute: query.
    ^self removeDuplicatesFrom: objects
```

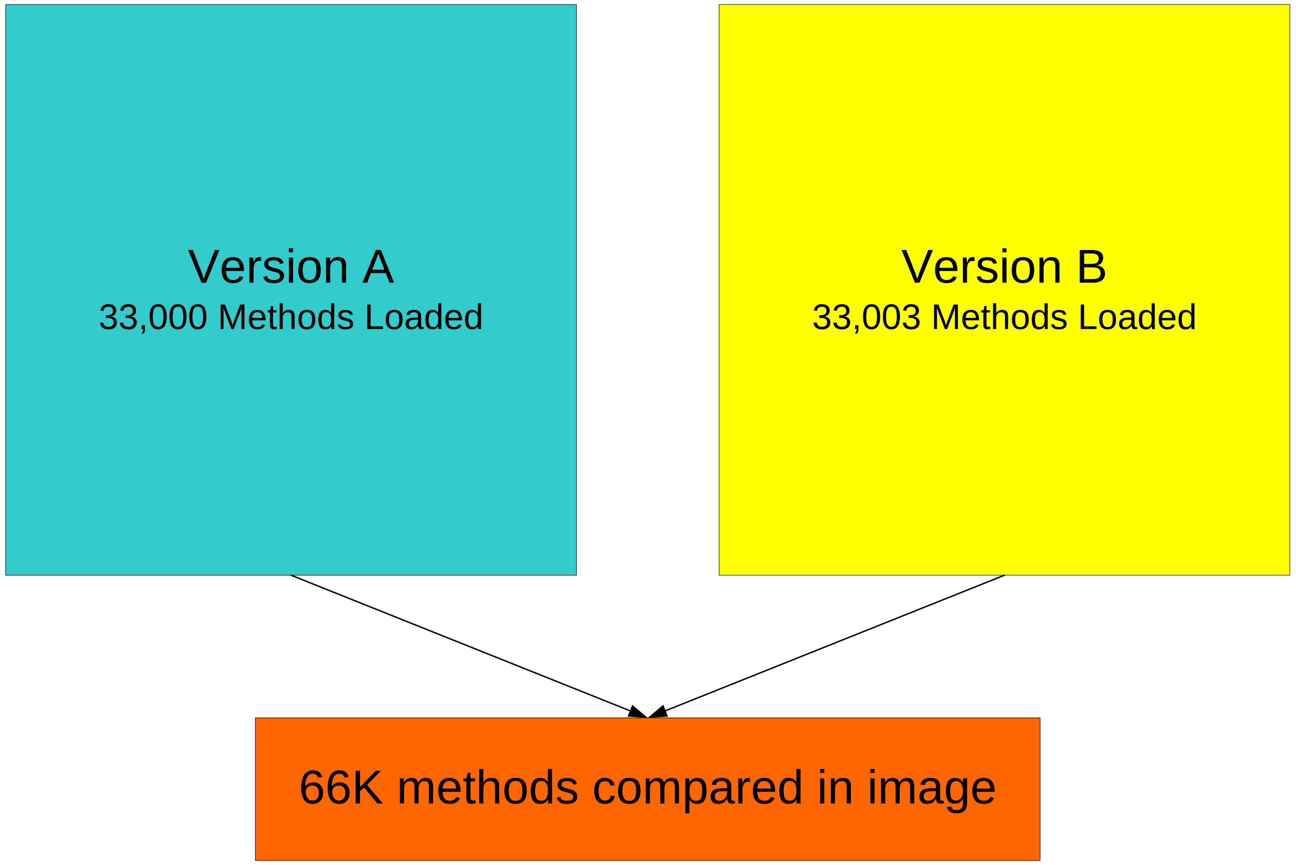
New Method Versions Query (VW7.10)

```
allVersionsWithName: aString inClass: aClassName in: aSession
    | query session |
    session := aSession ifNil: [StoreLoginFactory currentStoreSession].
    query := Query read: self
            where: [:eachLink | (eachLink definition name = aString)
                    AND: [(eachLink definition className = aClassName)
                    AND: [eachLink package timestamp = (
                        (Query readOneOf: self
                            where: [:eachLink2 |
                                eachLink2 definition id = eachLink definition id])
                            retrieve: [:eachLink2 | eachLink2 package timestamp min];
                            yourself)]]].
    query alsoFetch: #definition.
    query alsoFetch: [:each | each definition source].
    query alsoFetch: [:each | each package].
    query orderBy: [:each | each definition timestamp descending].
    query orderBy: [:each | each package timestamp].
    ^(session execute: query) asOrderedCollection
```

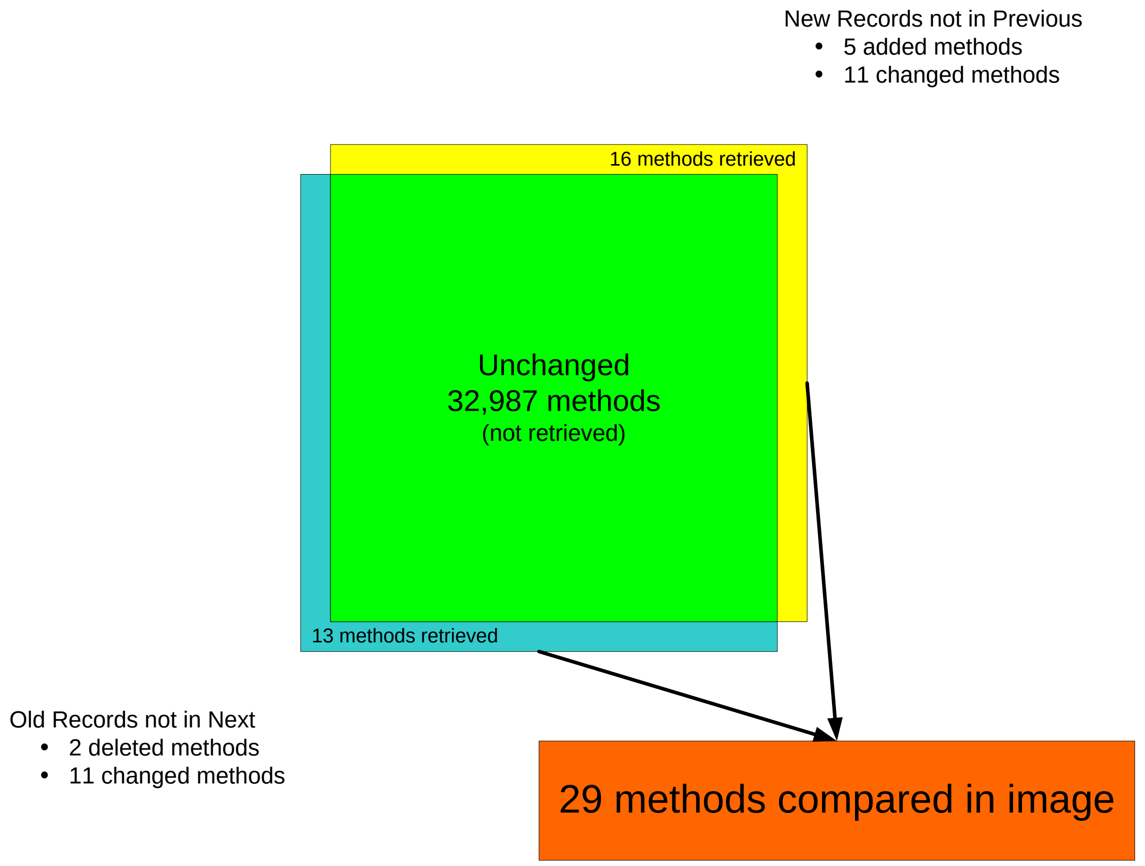
Query 2: Improving Package Comparison (VW 7.9)

- Original method:
  - Load version A
  - Load version B
  - Match up equivalent objects
  - Compare and show differences
- New method
  - Load different database records only
  - Match up equivalent objects
  - Compare and show differences

In-Memory Method Comparison



Differential Retrieval



Differential Retrieval Code

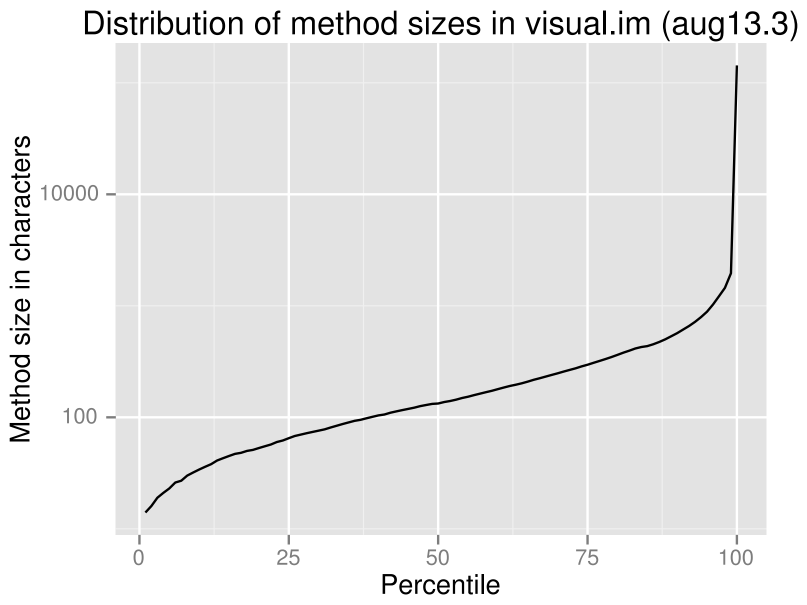
```
allMethodDifferencesWith: anotherStorePackage
      | keys query1 query2  union |
      keys := Array with: self primaryKey with: anotherStorePackage primaryKey.
      query1 := Query read: StoreMethodInPackage
        where: [:each |   |subQuery |
                subQuery := Query read: StoreMethodInPackage where: [:foo |
                                foo packageRef = keys last].
                subQuery retrieve: [:x | x methodRef].
                (each packageRef = keys first)
                                & ((each methodRef) notIn: subQuery)].
      query1 alsoFetch: [:each| each definition].
      query1 alsoFetch: [:each| each definition source].
      query1 alsoFetch: [:each| each package].
      query2 := Query read: StoreMethodInPackage...
      …
      union := query1 unionAll: query2.
      union requiresDistinct: false.
      ^session execute: union.
```

SCHEMA

Possible Schema Changes

- Bundle/Package version globally unique identifiers
- Modifying method source storage
  - Currently using chained blobs
  - Modify to use a character format, possibly with special provision for very large methods
- Modifying file storage
  - Also use chained blobs
  - Modify to use blocked blobs on Oracle
- These are ideas which may not survive testing

Method Source Storage (Currently 32k)



Percentile	Method size
90.0	569
99.0	1957
99.5	3067
99.9	6835

Needed To Support Schema Changes

- Changes have to work across databases
  - Taking advantage of database specific features
  - Dealing with database shortcomings
  - Occaisionally may require Glorp enhancements
- Solutions for customers
  - With multiple Cincom® ObjectStudio® or Cincom® VisualWorks® versions
  - With large existing repositories

Questions? Comments?

Ask now

--- or ---

Tom Robinson
trobinson1@cincom.com

Contact Information

- **Suzanne Fortman** (sfortman@cincom.com)
  Cincom Smalltalk Program Director
- **Arden Thomas** (athomas@cincom.com)
  Cincom Smalltalk Product Manager
- **Jeremy Jordan** (jjordan@cincom.com)
  Cincom Smalltalk Marketing Manager

http://www.cincomsmalltalk.com