SMALLTALK SOLUTIONS TORONTO APRIL 2006

Silt: Lessons Learned in a Smalltalk Web Deployment

Wednesday, September 6, 2006

How to Scale a Smalltalk Server Without Any Planning

James A. Robertson
Product Manager
Smalltalk
Cincom Systems, Inc.

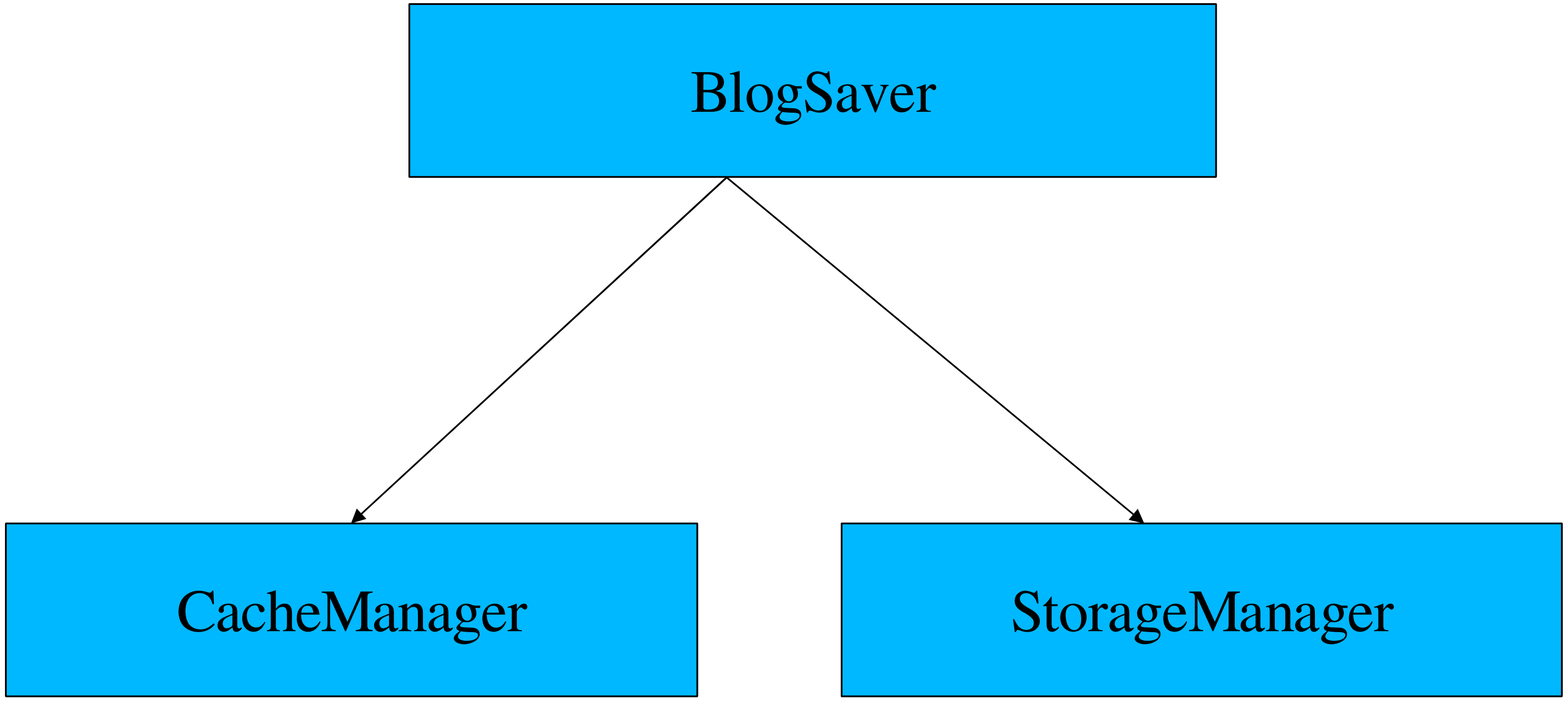
Agenda

- The Server: Basic Architecture
- A Few problems
- Summary

Project Discussed

- Silt
  - http://www.cincomsmalltalk.com/CincomSmalltalkWiki/Silt
  - http://www.cincomsmalltalk.com/blog/blogView
- Managed in the public Store
  - Silt is public domain

Architecture



Architecture

- BlogSaver
  - The “well known” API point for the server
  - Originally, it was the entire server
  - It still has way too much code in it ☺
  - One instance per blog

Architecture

- StorageManager
  - Manages the storage and retrieval of posts
  - Extracted out of the BlogSaver class
  - One serialized object file per day
  - Posts (and their comments) are in a collection in that object file

Architecture

- CacheManager
  - Holds cache for the server
    - Entire main page
    - Last N individual posts asked for
    - Keyword search cache
    - Category search cache
    - Dictionary of posts by year
      - Older posts are less likely to change

Architecture

- Initially, BlogSaver was it
  - Singleton
  - Assumed a single blog
  - Lots of references to it in the servlets, etc.

Problems

- First problem: Multiple Blogs
  - I had set up the ability to have multiple posters
  - I had not set up for multiple blogs
  - Michael Lucas-Smith broached the subject
    - I think he thought the delay was legal
    - It was actually inertia – I didn’t want to do the work!

Problems

```
Smalltalk.Blog defineClass: #AbstractBlogSaver
    superclass: #{Core.Object}
    indexedType: #none
    private: false
    instanceVariableNames: 'users settings ipFileSem settingsFile
    syndicationSem '
    classInstanceVariableNames: 'default '
    imports: ''
    category: 'Blog‘
```

Key was the “default” class instance variable

Problems

- BlogSaver named: ‘someName’.
  - The class instance variable holds a dictionary of blog instances
  - Those are created from configuration files
  - Allowed me to set up multiple blogs
  - There are now 24 active blogs, and a few inactive ones
  - Could easily add new Smalltalk servers and segregate by blog

Problems

- Second Problem: Dynamic Request Backup
  - Posts are stored “one file per day, all posts in that file”
  - To get the last few posts, every request ended up reading the same files repeatedly

Problems

- Solution: Added a simple cache of all the posts that belong on the front page
  - New requests simply return the cached data
  - Cleared out on updates to relevant posts, or on new posts
  - Immediately made the blog more responsive

Problems

- Third Problem: Slow Category Searches
  - Each post can have a category
  - Category searches required a scan of all posts
  - Fine at first, but… I’ve been at this since 2002

Problems

- Solution: A simple cache
  - This is when I split out the CacheManager class
  - One per blog
  - Holds a Dictionary, where the keys are the categories, and the values are the set of files containing matching posts
  - One time hit to populate, updated on each new post or update
  - Cache is saved to disk, so it does not need to be recreated at startup

Problems

- Speeded up category searches tremendously
  - Only have to open matching files
  - Linear search for matching posts in files
  - “fast enough”
  - Considering Ajax for caching large result sets

Problems

- Fourth Problem: Keyword Searches
  - Same problem as category searches, but cannot do full up front cache
  - Built same solution
  - Cache the results as they get queried
  - Still wasn’t fast enough

Problems

- The issue: Scanning all blog posts in the process that got kicked off by the servlet
  - Runs at same priority as other queries
  - Bogged the server down with I/O and CPU demands

Problems

- Solution: Class Promise
  - Blogged: http://www.cincomsmalltalk.com/blog/blogView?showComm=true&entry=3307882025

Problems

- Original Code:

```
allResults := self actuallySearchFor: searchText
                    inTitle: searchInTitle
                    inText: searchInText.
^allResults asSortedCollection: [:a :b | a timestamp > b timestamp].
```

- New Code:

```
promise := [self actuallySearchFor: searchText
                                inTitle: searchInTitle
                                inText: searchInText] promiseAt: Processor
    userBackgroundPriority.
allResults := promise value.
^allResults asSortedCollection: [:a :b | a timestamp > b timestamp].
```

Problems

- The Promise executes in the background, and the asking thread waits as it executes
- Allows other server threads to execute
- Extended Back to Category searches
- As with Category searches, considering an Ajax solution

Problems

- Still expensive: reading all posts takes time
- Added a cache for posts, keyed to year
  - Older posts unlikely to change
  - Flush cache for year on change
  - Makes searches much faster

Problems

- Fifth Problem: Spam
  - Comments
  - Trackbacks
  - Referers

Problems

- In the server, comments and trackbacks are handled the same way – i.e., solve one, solve both
- Referers are gleaned from the server logs

Problems

- Comments/Trackbacks
  - Turned off comments on posts off the front page
  - Added a "no more than N hrefs" rule for comments
  - Added an IP throttle
- These steps mostly ended comment spam
- Turned off Trackback – it's a spam garden

Problems

- Referer Spam
  - Bogus referrals from porn/pharma/etc sites
  - Added a constantly updated blacklist of keywords
  - List is updated every few hours

Problems

- The referral scanner was eating the server!
  - Executing the scan over the logs for each of the blogs was wasteful
  - Unified the scan
  - Still ate too much time
  - Ended up extracting the process from the server, set it up as a CRON job
  - The blog instances just look for (and cache) the referral file every few hours

Summary

Summary

- I only solved these problems as they came up
  - I had no idea that they would be problems ahead of time
- I patch the server live
  - Update the code on the fly, including shape changes to classes.

Summary

- I’ve yet to hit a problem that wasn’t my fault
- Smalltalk is a powerful, scalable solution for web applications

Contact Info

- James Robertson
  - Jarober@gmail.com
  - Jrobertson@cincom.com
- Silt
  - http://www.cincomsmalltalk.com/CincomSmalltalkWiki/Silt
- BottomFeeder
  - http://www.cincomsmalltalk.com/BottomFeeder