Scaling Applications:
Introduction to GemStone

James Foster – ESUG 2010

Barcelona, Spain

About your presenter…

- James Foster
  - On the Smalltalk Team at VMware
  - Programming since 1971
  - Also, former attorney and commercial pilot

Agenda

- Smalltalk
  - Influence, history, image-based development
- GemStone
  - Overview
  - Global namespaces
  - Class Versions
  - GLASS
  - Getting started

History

- Smalltalk originated in the 1970s
  - Each hardware had its own operating system
  - Language tended to be tied to host environment
  - Tools tied to environment

Not: "file-based" development

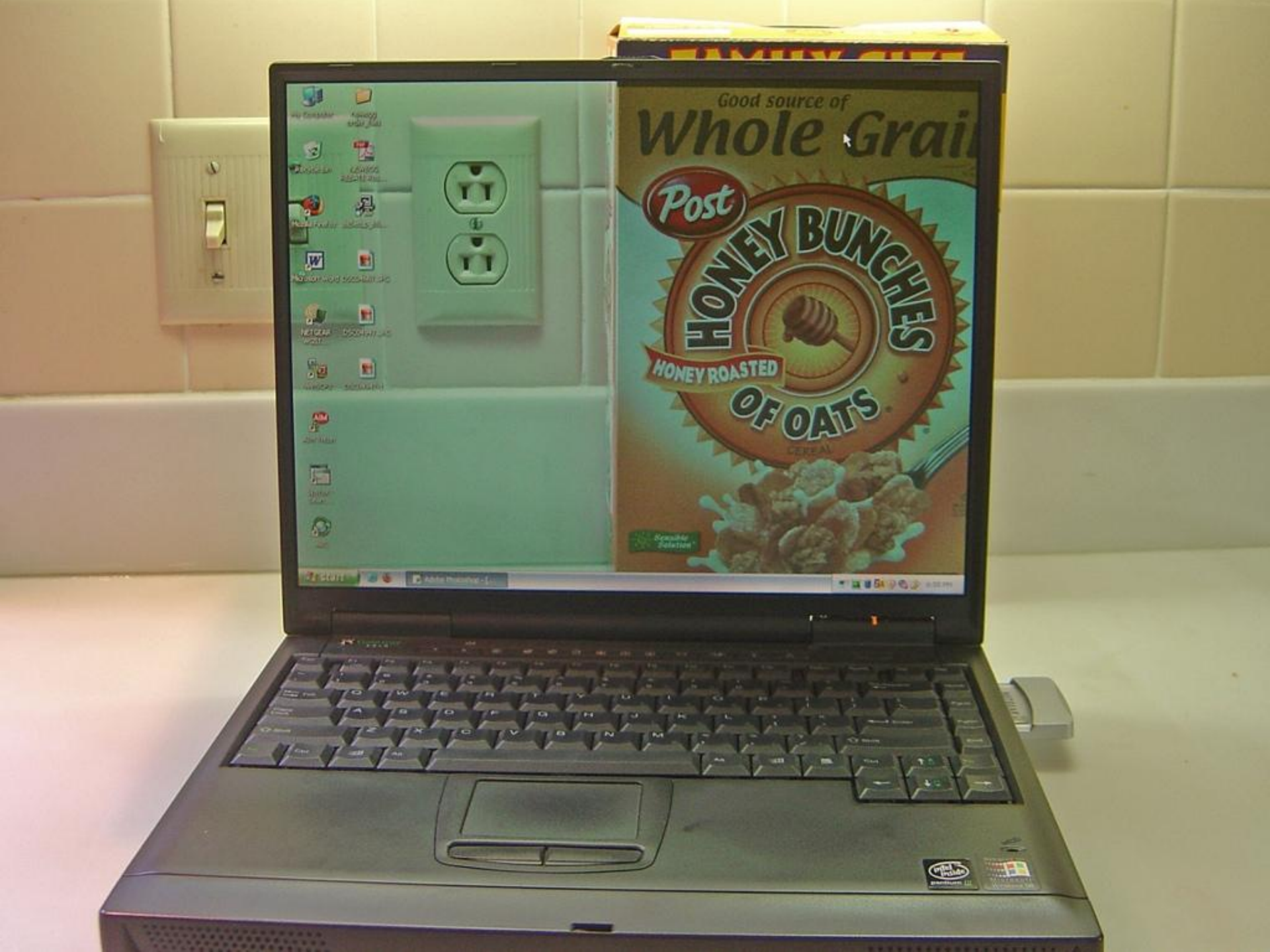
- Characteristics of "file-based" development
  - Schema and code stored in text files external to the system under construction
  - Compiled by a separate program into a new executable file
    - Alternate: execution at runtime by an interpreter
  - Persistent data is external to the program
  - "Integrated Development Environment (IDE)" merges tools

Analogy: DBMS

- Database Management System (DBMS)
  - Schema and code stored internally with data
    - External text files are secondary and optional
  - External tools available to manipulate system
  - Interaction with immediate feedback on current state of system
  - Incremental changes take system immediately from one state to another
  - Initial state has some built-in capability

Analogy: spreadsheet

- Spreadsheet
  - Single tool for data, code, editing, executing, etc.
  - Modifications are preserved if environment is saved
  - Interaction with immediate feedback on current state of system
  - Incremental changes take system immediately from one state to another
  - Initial state has some built-in capability


Good source of
Whole Grai
Post
HONEY BUNCHES
HONEY ROASTED
OF OATS
CEREAL

Analogy: (virtual) computer

- Computer
  - Modifications are made from inside the environment
  - Persistent (saved) shutdown/restart
- Compare to virtual hardware (VMware, Parallels, Xen, etc.)
- Complete, self-contained environment

“Image-based” development

- Smalltalk is a programming environment (not merely a language)
  - An **Object** represents a combination of behavior and properties (code and data)
    - Each object is an instance of a **Class**
  - **Object Space** represents RAM
  - **Virtual Machine** represents CPU
  - **Image** represents disk
    - Persist copy of memory when system is not active

All-inclusive environment

- Open an “image” and step into a magical world
- Develop by modifying an existing environment
  - New environment created by copying (cloning)
- Environment contains all your tools
  - Schema definition tools, source code editor, runtime code debugger, object inspector, source code management, etc.
- Save an “image” to create a snapshot

Advantages of Smalltalk

- Pure object system
  - "Everything" is an Object
- Simple, elegant language
- Powerful class libraries

Limitations of traditional Smalltalks

- Object space (image) must fit in (virtual) RAM
- Object space visible to only one VM
- Sharing objects between VMs is difficult
  - Convert to non-object format (binary, XML, SQL)
  - No built-in object identity for multiple exports
- Object state is lost when VM exits

Welcome to the magical world of GemStone

- Object space limited by disk, not RAM
- Object space shared across multiple VMs on multiple hosts
- Transactional persistence

- Image from http://www.flickr.com/photos/laffy4k/182219003/
- Creative Commons License http://creativecommons.org/licenses/by-sa/2.0/

Intro: What is GemStone/S?

- A Smalltalk environment?
- A database system?

- Two in one!

GemStone

- is a multi-user object server
- is a programmable server object system
- manages a large-scale repository of objects
- supports partitioning of applications between client and server
- supports queries and indexes for large-scale object processing
- supports transactions and concurrency control in the object repository
- supports connections to outside data sources
- provides login security and account management
- provides services to manage the object repository
- provides comprehensive statistics and charting for performance tuning

Multi-User Object Server - 1

- Scalable
  - Thousands of concurrent user sessions
  - VMs on hundreds of hosts
  - Object space (image) of terabytes
  - Object count of 2^40 (~trillion)
  - Thousands of transactions per second

Multi-User Object Server - 2

- Concurrency
  - Multiple user sessions can be active
  - Each user may have multiple sessions
  - Separate or shared namespaces per user
  - Changes to objects are committed in transaction
  - Concurrency controls and locks for coordination

Multi-User Object Server - 3

- User-based Security
  - Login (password)
  - Namespace: global lookup visibility
  - System operations (backup, change password, …)
  - Per-object read/write access by user/group

Programmable Object Server

- Smalltalk
  - Data definition
  - Object manipulation
  - Query facilities
  - Concurrency management
  - System management

Client/Server Partitioning

- GemStone C Interface (GCI)
  - API to login, manipulate objects, send messages
- GemBuilder for Java
  - Java library wrapping the C library
- GemBuilder for Smalltalk
  - Smalltalk library wrapping the C library
  - Transparent replication and synchronization
  - Preserves object identity across multiple fetches

Limitations of traditional Smalltalks

- Object space (image) must fit in (virtual) RAM
- Object space visible to only one VM
- Sharing objects between VMs is difficult
  - Convert to non-object format (binary, XML, SQL)
  - No built-in object identity for multiple exports
- Object state is lost when VM exits

Large-Scale Object Space

- Object space limited only by disk size
  - System handles RAM caching
- Object space shared by all connected sessions
  - View based on point of last commit/abort
- Persistence by reachability from root object
  - Attach new objects to an existing collection
- On commit, new & changed objects are visible
  - Other sessions must commit/abort to see changes

Indexes

- UnorderedCollections (Set, …) can be indexed

  `#'createEqualityIndexOn:withLastElementClass:'`

  – This creates a B-Tree
- Index maintenance is automatic
  – Adding or removing from the collection
  – Changing an instance variable of a member object

Queries

- Special syntax to query indexed collections

```
subSet := set select: {:each |
  each.surname = 'FOSTER' & each.givenname = 'JAMES'}.

stream := set selectAsStream: {:each |
  each.surname >= 'FOSTER'}.
list := stream next: 10.
```

  - Allows very efficient 'VCR' widget

Transaction Semantics

- Database view is ***isolated*** with ***repeatable reads***
  - Changes made before a commit are not shared
  - Committed changes made by others since current view was obtained are not visible until current session does an abort/commit
  - On abort the current view is reset to current state, including any persistent objects for which changes were made and abandoned

Optimistic Concurrency

- Wait till commit attempt to check conflicts
  - When a commit is requested, system checks to see if any objects modified in current session have been modified by another session after this view
  - If there are any ***write-write conflicts***, then commit fails and view updates to most recent (except for changed objects)
  - Current session must abort, get a fresh view, and choose how to proceed

Pessimistic Concurrency

- At any time any session may request a ***write-lock*** on an object
- Once a write-lock is issued, no other session can commit a change to the locked object
  - To be sure, do an abort/commit after locking
- Now proceed to make changes!
- API exists to query system for lock owner

Reduced-Conflict Classes

- Sometimes it is acceptable to have two sessions make well-defined changes to the same object at the same time
  - Parallel sessions add a new object to a Set
- Rc* classes have safe, well-defined semantics
  - RcIdentityBag
  - RcKeyValueDictionary
  - RcQueue
  - RcCounter

Interface to External Systems

- Write a ***UserAction*** in C and load it into VM
  - Invoke function in UserAction and pass object
  - Get object back from function call
- GemStone add-on product called GemConnect
  - Smalltalk library and C-based UserAction library to interface with Oracle
- GemStone/S 64 Bit version 3 has a pure Smalltalk interface to C libraries

User Authentication

- Valid GemStone (database) user ID/password
  - Various configurable rules on passwords
  - Configurable limit on concurrent logins
- Optional: valid host OS user ID and password
  - Alternate is to set up to use a specified host user

User Authorization

- Privileged operations
  - Change own password
  - Change another's password
  - Compile Smalltalk code
  - Perform backups
  - Perform system-wide garbage collection
  - Access the file system (client and/or server)
  - Load/call a UserAction
  - Execute a shell command on the server

Object Security Policy

- Each user can be associated with one or more groups (e.g., Users, Administrators, etc.)
- Each object can be associated with one of up to 32767 security policies
- Security policy defines
  - Owner
  - Groups
  - Owner access (read/write)
  - Group access (read/write)
  - World access (read/write)

System Management

- On-line (live system) backup
- Database restore, including transaction logs
- Use shared memory to cache disk pages
- Use asynchronous I/O to parallelize disk writes
- Allow hosts to be added and removed
- Monitor and tune system for performance

Namespaces: Variable Binding

- Lookup when compiling a method:
  - Block arguments and temporaries
  - Method arguments and temporaries
  - Object instance variables
  - Class variables
  - Pool dictionaries
  - Globals

Namespace: Where are Globals? - 1

- Traditional Smalltalk implementations:
  - One SystemDictionary in which keys are global names
  - Exists in itself as value for key #'Smalltalk'
  - Happens to be root of object graph for GC

Namespace: Where are Globals? - 2

- GemStone implementation
  - Search an ordered list of dictionaries
  - List may be passed to the compiler
  - Each session has a default list based on the user
- Extra flexibility
  - Different methods can bind to different globals
  - Different users can share same dictionary
  - Additional complexity can be ignored

Root of Object Graph

- UserProfileSet (lookup with name #'AllUsers')
- UserProfile (userId symbolList …)
  - `System myUserProfile`
  - `AllUsers userWithId: #'DataCurator'`
- SymbolList (subclass of Array)
- SymbolDictionary
  - keys are Symbols; treated as global names
  - values are any objects
    - generally classes, singletons, and collections

Namespace Usage

- Deploy multiple applications in one database without name conflicts
- Compile application classes without visibility to development and testing classes
- Compile third-party library based solely on base classes
  - No name conflicts with application or other third-party libraries

Changing Class Schema - 1

- Traditional Smalltalk implementations
  - Save new class definition
  - Replace class as value in global dictionary
  - Find all instances of old class in the object space
  - Create new instance for each old instance
  - Copy values for instance variables with same names
  - Do #'become:' to swap old and new objects and preserve references
  - Garbage collect old instances and class definition

Changing Class Schema - 2

- GemStone complications
  - Large object space makes #'allInstances' expensive
  - Concurrency management prevents changing objects in another session's view

Changing Class Schema - 3

- Recall that even traditional approach does not really *change* class definition or objects
  - New class definition is created and installed
  - New instances are created to replace old ones
  - Illusion of change due to rapid replacement
- GemStone schema changes are gradual
  - Multiple class versions can co-exist
  - Objects are instances of some version of class

Changing Class Schema - 4

- Saving a class definition simply defines ***a*** class
- Definition may specify a previous version
  - New version may have different name
  - New version need not replace old global
- Two definitions must share a classHistory in order to migrate instances
- Instances may be migrated individually or as a group
- Custom migration methods can be written

User Interface (Tools)

- Topaz
- GemBuilder for Smalltalk (GBS)
- GemTools (Pharo)
- Others
  - Jade
  - WebTools
  - GsKit

GLASS

- Heard of LAMP (Linux, Apache, MySQL, Perl)?
- Try out GLASS:
  - **G**emStone
  - **L**inux
  - **A**pache
  - **S**easide
  - **S**malltalk
- Or: **G**emStone, **L**inux, **A**ida, **S**cribo, **S**malltalk

Getting Started

- No-cost license ("free as in beer" or "gratis")
  - Up to ~~4~~ 16 GB repository
  - ~~1~~ 2 GB shared page cache
  - ~~1~~ 2 CPU used (machine may have more)
- Virtual Appliance
  - Fully configured Linux system for VMware
- Native install on Mac or Linux

Conclusion

- Seaside Tutorial
  - http://seaside.gemstone.com/tutorial.html
- Resources (Download software and manuals)
  - http://seaside.gemstone.com/
- Questions?
  - JFoster@VMware.com
  - http://programminggems.wordpress.com