

Things I Wish I Knew about GemStone/S

ESUG 2011, Friday, 11:45 – 12:30

James Foster, Sr. Member Technical Staff

Inspiration

■ Nick Ager

- Suggested a session covering "the tools, common problems (eg empty statements, classHistory) backing-up and restoring, installation, Object Log, blocking vs Gem based servers, debugging, background processing etc."

■ Johan Brichau

- "Count me in too ;-)"

■ Stephan Eggermont

■ Norbert Hartl

- "Great idea. I'm in."

■ Diego Lont

■ Tobias Pape

- "/me rises his hand."

■ Conrad Taylor

Abstract

- **This presentation provides an introduction to GemStone/S, a multi-user Smalltalk with a built-in database. We briefly examine some issues observed by people who transition to GemStone/S from other Smalltalks.**
- **Depending on time, these topics may include installation, tools, backup/restore, class versions, debugging, concurrence, background processing, and repository-wide garbage collection.**
- **Caveat:**
 - There are three multi-day courses on GemStone/S.
 - 45 minutes is not enough time to cover any topic in depth!

■ Software Background

- As a junior-high student in 1971, I discovered the local university's computer center and a life-long obsession with computers began.
- I was introduced to Smalltalk/V for the Mac in the mid-90s, and became a Smalltalk bigot.
- I am on the Smalltalk Engineering team at VMware, and am a passionate advocate for GemStone and Seaside.

■ Past Careers

- Commercial Pilot
- Lawyer

■ Other interests

- Economics
- Politics
- Religion

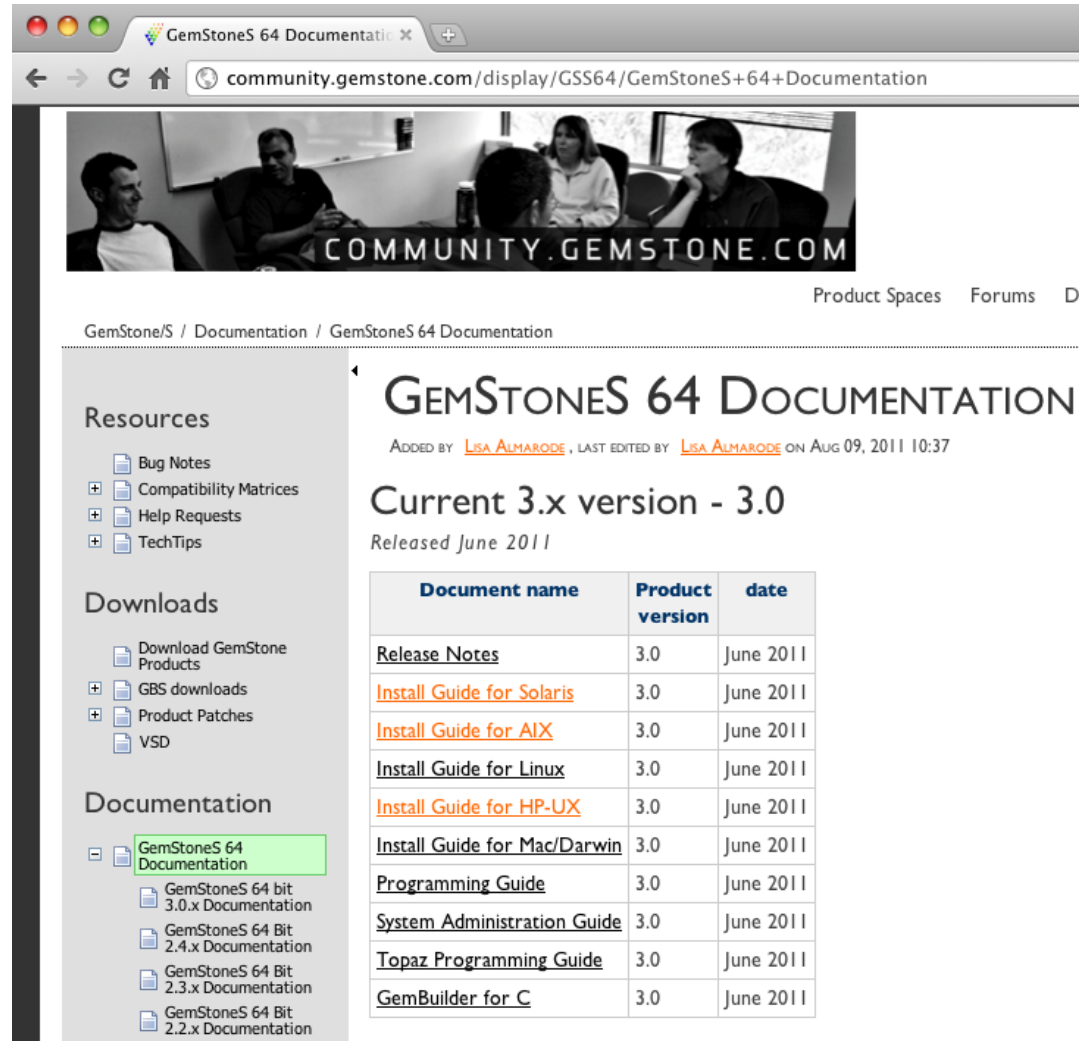
Agenda

- **Installation**
- Architecture
- Tools
- Backup/Restore
- Class Versions
- Debugging
- Concurrency
- Background Processing

Installing GemStone/S 64 Bit

Install Guide and Other Documentation

- <http://community.gemstone.com/display/GSS64/GemStoneS+64+Documentation>
- Release Notes
- Install Guides
 - Solaris
 - AIX
 - Linux
 - HP/UX
 - Mac/Darwin
- Programming Guide
- Systems Admin Guide
- Topaz Programming Guide
- GemBuilder for C



GemStone/S / Documentation / GemStoneS 64 Documentation

GEMSTONES 64 DOCUMENTATION

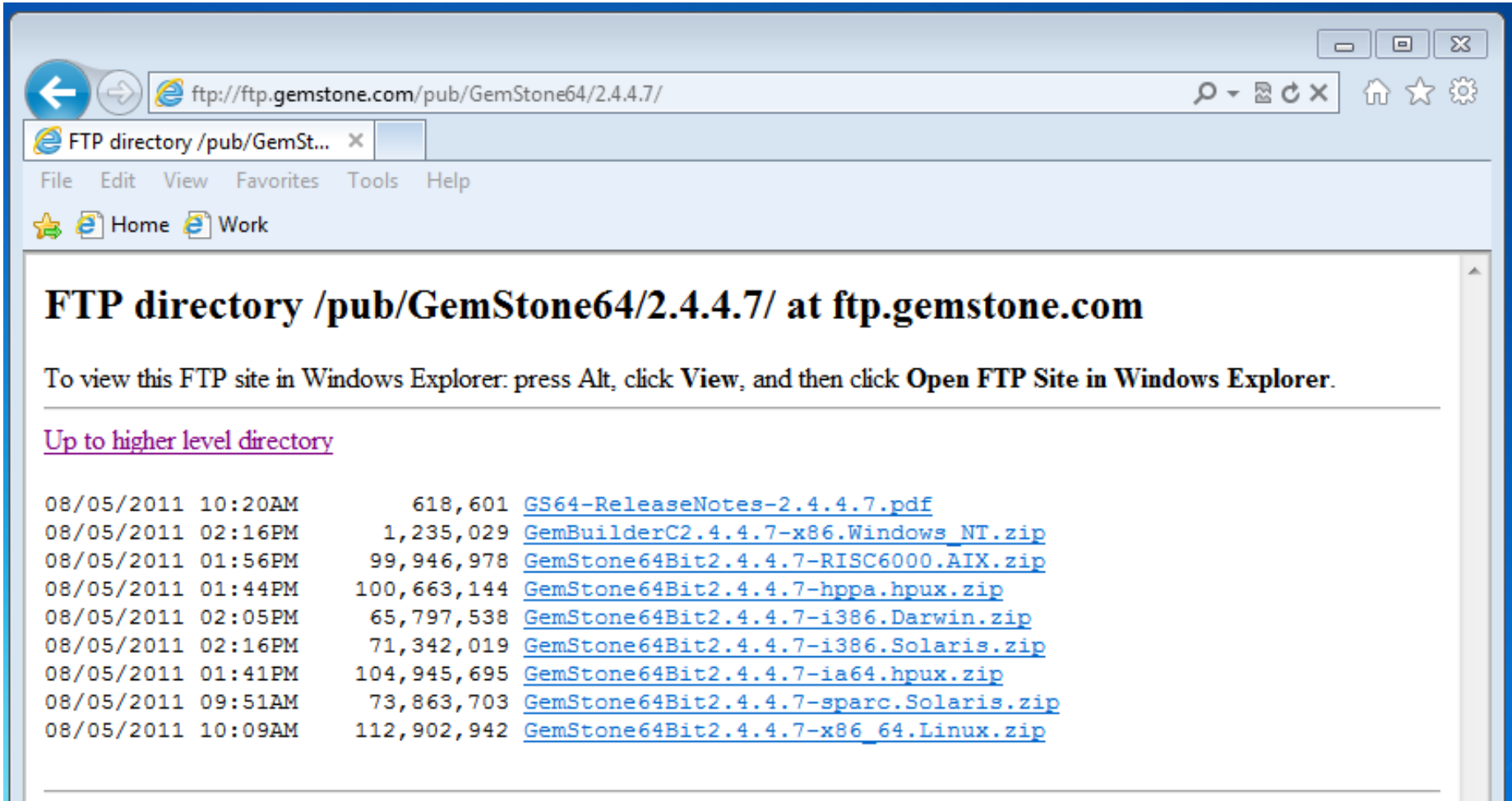
ADDED BY [LISA ALMARODE](#), LAST EDITED BY [LISA ALMARODE](#) ON AUG 09, 2011 10:37

Current 3.x version - 3.0
Released June 2011

Document name	Product version	date
Release Notes	3.0	June 2011
Install Guide for Solaris	3.0	June 2011
Install Guide for AIX	3.0	June 2011
Install Guide for Linux	3.0	June 2011
Install Guide for HP-UX	3.0	June 2011
Install Guide for Mac/Darwin	3.0	June 2011
Programming Guide	3.0	June 2011
System Administration Guide	3.0	June 2011
Topaz Programming Guide	3.0	June 2011
GemBuilder for C	3.0	June 2011

Product Distribution

- <ftp://ftp.gemstone.com/pub/GemStone64/2.4.4.7>



The screenshot shows a web browser window displaying an FTP directory listing. The address bar shows the URL <ftp://ftp.gemstone.com/pub/GemStone64/2.4.4.7/>. The browser's title bar reads "FTP directory /pub/GemSt...". The main content area displays the following text:

FTP directory /pub/GemStone64/2.4.4.7/ at ftp.gemstone.com

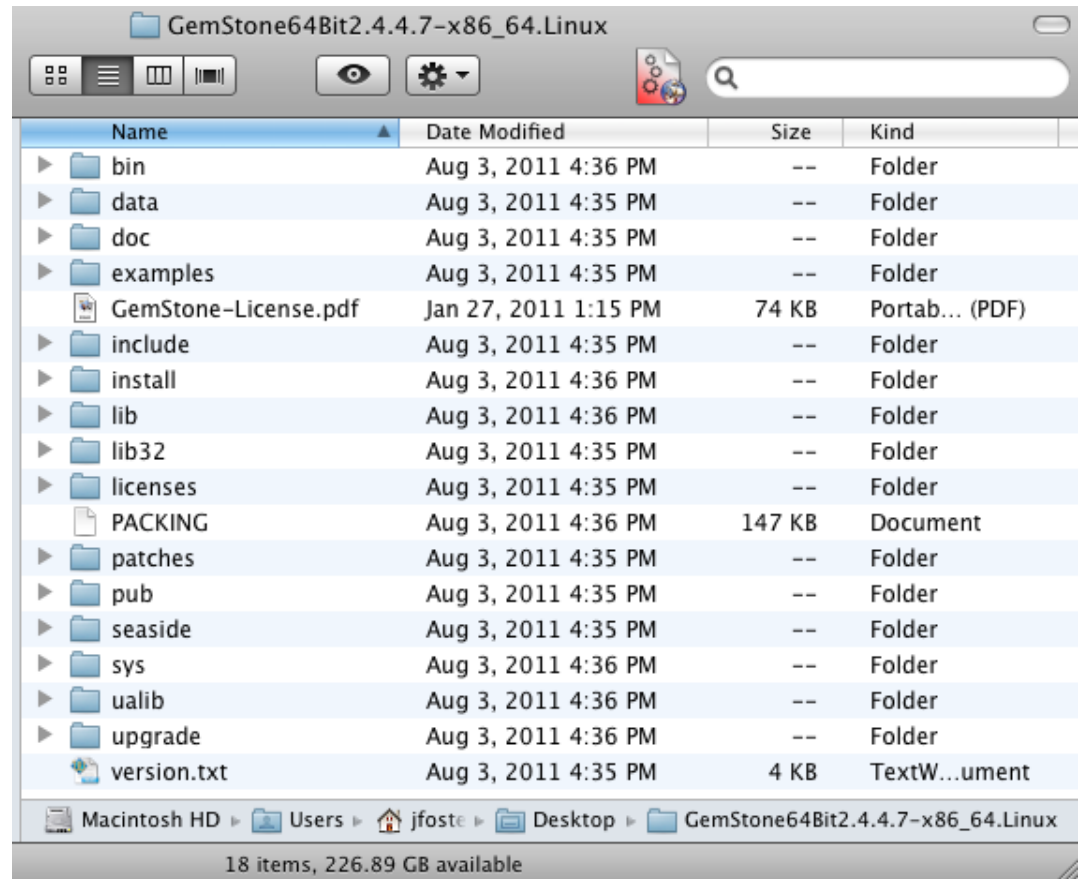
To view this FTP site in Windows Explorer: press Alt, click **View**, and then click **Open FTP Site in Windows Explorer**.

[Up to higher level directory](#)

08/05/2011 10:20AM	618,601	GS64-ReleaseNotes-2.4.4.7.pdf
08/05/2011 02:16PM	1,235,029	GemBuilderC2.4.4.7-x86.Windows_NT.zip
08/05/2011 01:56PM	99,946,978	GemStone64Bit2.4.4.7-RISC6000.AIX.zip
08/05/2011 01:44PM	100,663,144	GemStone64Bit2.4.4.7-hppa.hpux.zip
08/05/2011 02:05PM	65,797,538	GemStone64Bit2.4.4.7-i386.Darwin.zip
08/05/2011 02:16PM	71,342,019	GemStone64Bit2.4.4.7-i386.Solaris.zip
08/05/2011 01:41PM	104,945,695	GemStone64Bit2.4.4.7-ia64.hpux.zip
08/05/2011 09:51AM	73,863,703	GemStone64Bit2.4.4.7-sparc.Solaris.zip
08/05/2011 10:09AM	112,902,942	GemStone64Bit2.4.4.7-x86_64.Linux.zip

Suggested Install Location

- **/opt/gemstone/** # root for GemStone-related things
 - /opt/gemstone/<productDirectory>/
 - Add symbolic link from /opt/gemstone/product
 - /opt/gemstone/backups
 - /opt/gemstone/bin
 - /opt/gemstone/data
 - /opt/gemstone/etc
 - /opt/gemstone/locks
 - /opt/gemstone/log
 - /opt/gemstone/Monticello
 - /opt/gemstone/product
 - \$GEMSTONE points here



Operating System Configuration

■ Shared Memory settings in `/etc/sysctl.conf`

■ Linux:

- `kern.shmmax = nnnn` # max shared memory segment in bytes
- `kern.shmall = nnnn` # max 4096-byte pages for shared memory

■ Macintosh:

- `kern.sysv.shmmax = nnnn` # max shared memory segment in bytes
- `kern.sysv.shmall = nnn` # max 4096-byte pages for shared memory

Environment Variables

- **See System Administration Guide, Appendix E for full list**
 - GEMSTONE # full path to product
 - /opt/gemstone/product
 - GEMSTONE_EXE_CONF # dir or file for executable config files
 - /opt/gemstone/etc
 - GEMSTONE_NRS_ALL # settings for default directory, log files, etc.
 - #dir:/opt/gemstone#log:/var/log/gemstone/%N_%P.log
 - GEMSTONE_SYS_CONF # dir or file for system-wide config file
 - /opt/gemstone/etc/system.conf
 - upgradeLogDir # used for upgrade to new GS/S version
- PATH should include \$GEMSTONE/bin

Keyfile Capabilities

- **Use of GemStone/S 64 Bit requires a "keyfile" that identifies allowed capabilities (with "Web Edition" no-cost license limits)**
 - Max repository size (unlimited)
 - Max object count (unlimited)
 - Max concurrent logins (unlimited)
 - Expiration date (none)
 - Machine type (Linux or Macintosh)
 - Max shared page cache (2 GB)
 - Max CPUs used (2)
 - Allow use of traversal buffer in GCI-to-Gem communications (no)
 - Required for use of GemBuilder for Smalltalk (for VA Smalltalk and Cincom Smalltalk)
 - Allow Gems on non-Stone machine (no)

Keyfile Location

- **\$GEMSTONE/seaside/etc/gemstone.key** is Web Edition keyfile
 - Also available from <http://seaside.gemstone.com/etc/>
- **Location specified in config file to override default**
 - KEYFILE = \$GEMSTONE/sys/gemstone.key

Config File

- **Default is at `$GEMSTONE/data/system.conf`**
 - `GEMSTONE_SYS_CONF` environment variable specifies another location
- **Four (4) required configurations (included in default file)**
 - `DBF_EXTENT_NAMES = $GEMSTONE/data/extent0.dbf;`
 - `STN_TRAN_FULL_LOGGING = FALSE;`
 - `STN_TRAN_LOG_DIRECTORIES = $GEMSTONE/data/, $GEMSTONE/data/;`
 - `STN_TRAN_LOG_SIZES = 100, 100;`
- **Suggest a file such as `/opt/gemstone/etc/seaside.conf`**
 - Include only non-default configurations
 - Above plus `KEYFILE`

Agenda

- Installation
- **Architecture**
- Tools
- Backup/Restore
- Class Versions
- Debugging
- Concurrency
- Background Processing
- Repository-wide Garbage Collection

Architecture: What is Different about GemStone?

GemStone Enhancements over Typical Smalltalk

- **Large object space**
 - Object space is (in practice) limited only by disk (not by RAM)
- **Transactions**
 - Related updates can be grouped in an “all-or-nothing” transaction
- **Persistence**
 - Transactions are immediately recorded to a log file
- **Multi-user**
 - Thousands of virtual machines can interact with a single object space
- **Multi-machine**
 - Virtual machines can be on hundreds of hosts

Complexities

- **Large object space**
 - Disk is slow, so cache recently-used objects in RAM
- **Transactions**
 - Group changes to support roll-back (abort)
- **Persistence**
 - Recover recent changes in event of crash
- **Multi-user**
 - Isolate each user's view (repeatable read)
 - Manage concurrency conflicts (avoid simultaneous updates to same object)
 - Manage object and class versions (when are updates visible to others?)
- **Multi-machine**
 - Coordinate object updates between machines

Programming Issues

■ Garbage collection (GC)

- Temporary objects local to virtual machine
- Persistent objects in shared object space

■ Large collections

- Iterating can be slow
- Use indexes to improve performance

■ Transactions

- Maintaining obsolete views can be expensive
- Object versions (different views of same object can see different values)
- Class versions (schema updates are not immediately applied to objects)
- Avoiding unnecessary concurrency conflicts

Architecture: How it is Done

GemStone Architecture

■ Repository

- Disk-based “image” holds objects and other data
- Made up of **extents** (files or raw partitions)
- Objects are on 16k **pages**

■ Gem Process(s)

- Single Smalltalk virtual machine

■ Stone Process

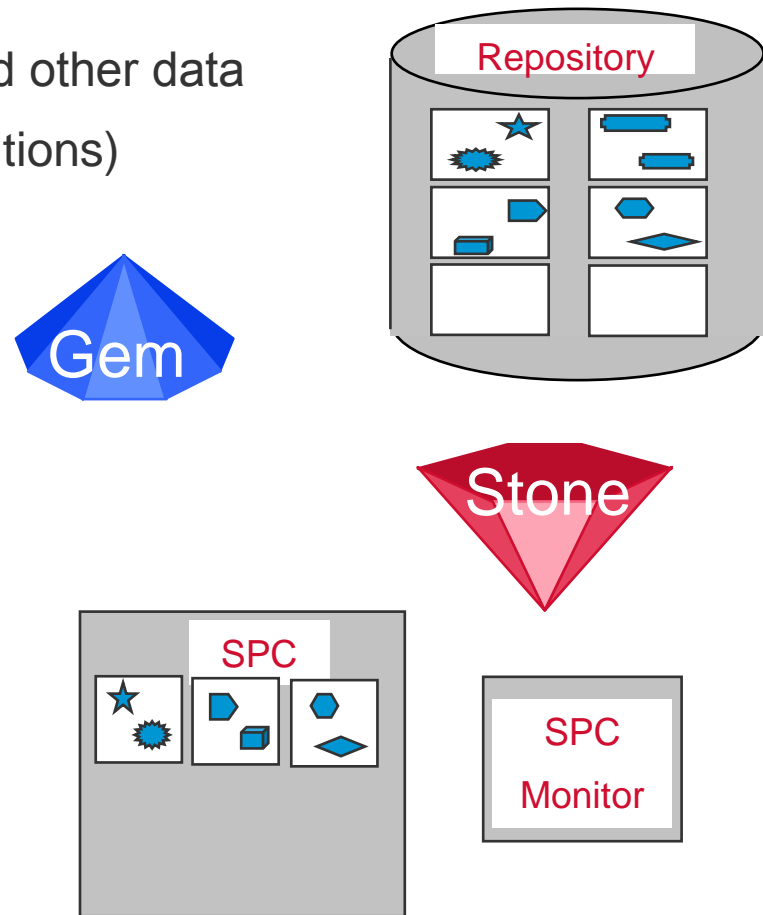
- Manages concurrency

■ Shared Page Cache

- Fast cache of pages from repository
- Managed by SPC Monitor process

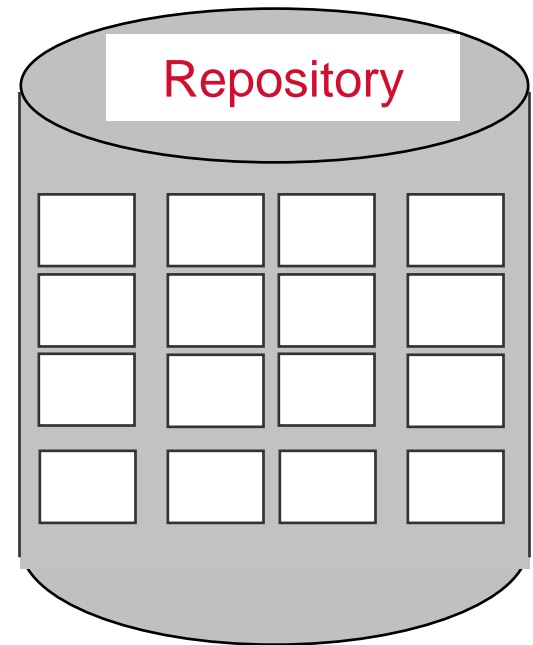
■ Other Processes

- GC Gems, Symbol Gem, AIO Page Server, Free Frame Server, etc.



Repository and Extent(s)

- Holds persistent GemStone objects (i.e., the “image”)
- Made up of 1 to 255 *extents* of up to 32 terabytes each
 - On-demand grow
 - Pre-grow to maximum size
- Each extent is composed of 16 KB *pages*
 - Root Pages
 - Object Table Pages
 - Data Pages
 - Commit Record Pages
 - Free OID List
 - Free Page List
- Page ID designates extent and offset
- Statistics: FreePages (Gem vs. Stone)



System Startup

■ 'startstone' command

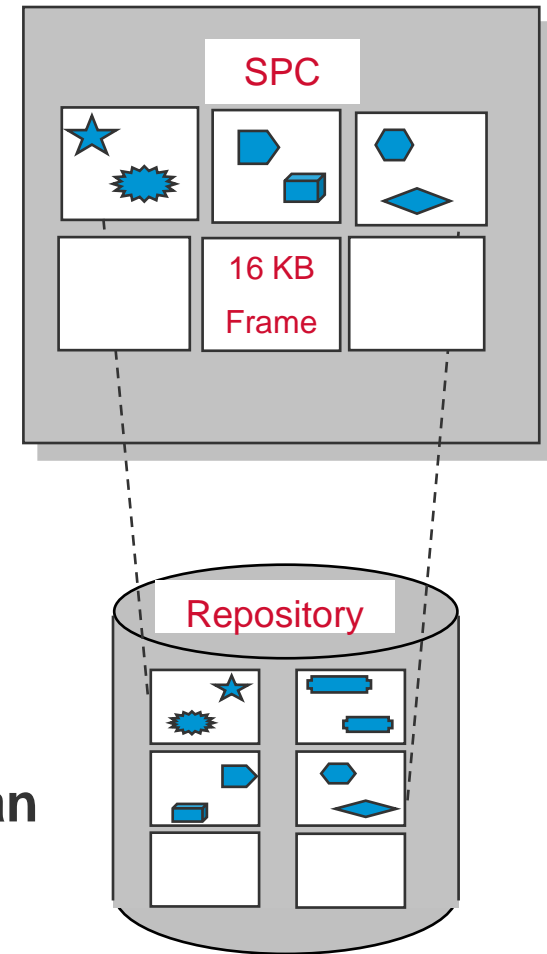
- Command line arguments for database name and other configurations
- Finds and opens all extents specified in required config file
- Finds and opens transaction log specified in required config file
- Starts Shared Page Cache (SPC) Monitor process (which allocates SPC)
- Starts other processes
 - AIO Page Server(s)
 - Free Frame Server(s)
 - Symbol Gem
 - GC Admin Gem
 - Reclaim Gem(s)
- Restores missing transactions if last shutdown was not clean (i.e., crash)
- Waits for requests from Gems (login, lock, commit, etc.)

Shared Page Cache

- Typical database challenge: disk is slow
- In-RAM cache of pages from repository
- Gem(s) may “attach” (or lock) in-use *Frames*
- Frame may contain a “dirty” page
- Async IO Page Server(s) write to repository
- Reuse frame only if it is unattached and clean
- Free Frame List
 - Maintained by SPC Monitor
 - Might be incomplete
 - Gem might be forced to scan cache
- Free Frame Server(s) scan for unattached & clean

Free Frame List

Frame #5
Frame #7
Frame #8



Shared Page Cache Statistics

■ Shrpc

- FreeFrameCount
- GlobalDirtyPageCount
- LocalDirtyPageCount
- TargetFreeFrameCount
- TotalAttached

■ Pgsvr

- FramesAddedToFreeList

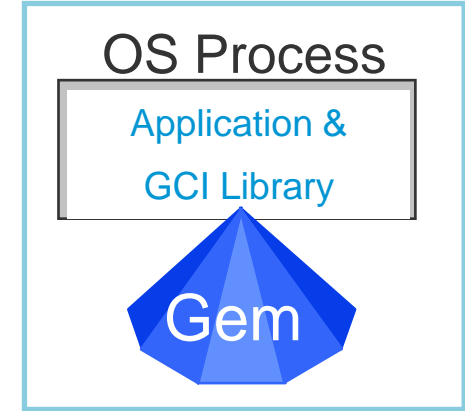
■ Gem, Stn

- AttachDelta
- AttachedCount
- FramesFromFindFree
- FramesFromFreeList
- LocalPageCacheHits
- LocalPageCacheMisses
- NonSharedAttached
- TimeInFramesFromFindFree

Gem Types

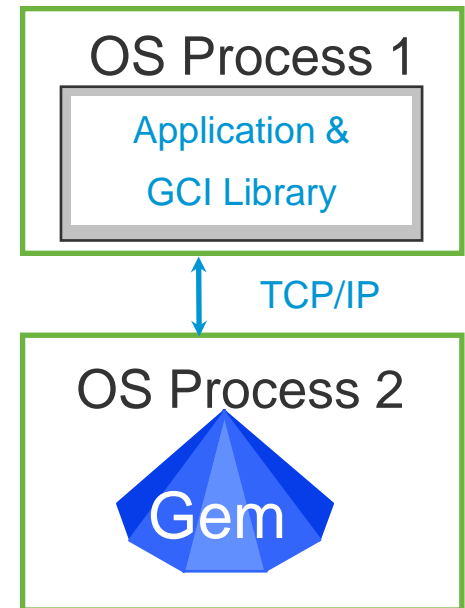
■ Linked Gem

- Application loads GemStone C Interface (GCI) library into its process space
- GCI library contains Gem code and runs in Application's OS process space



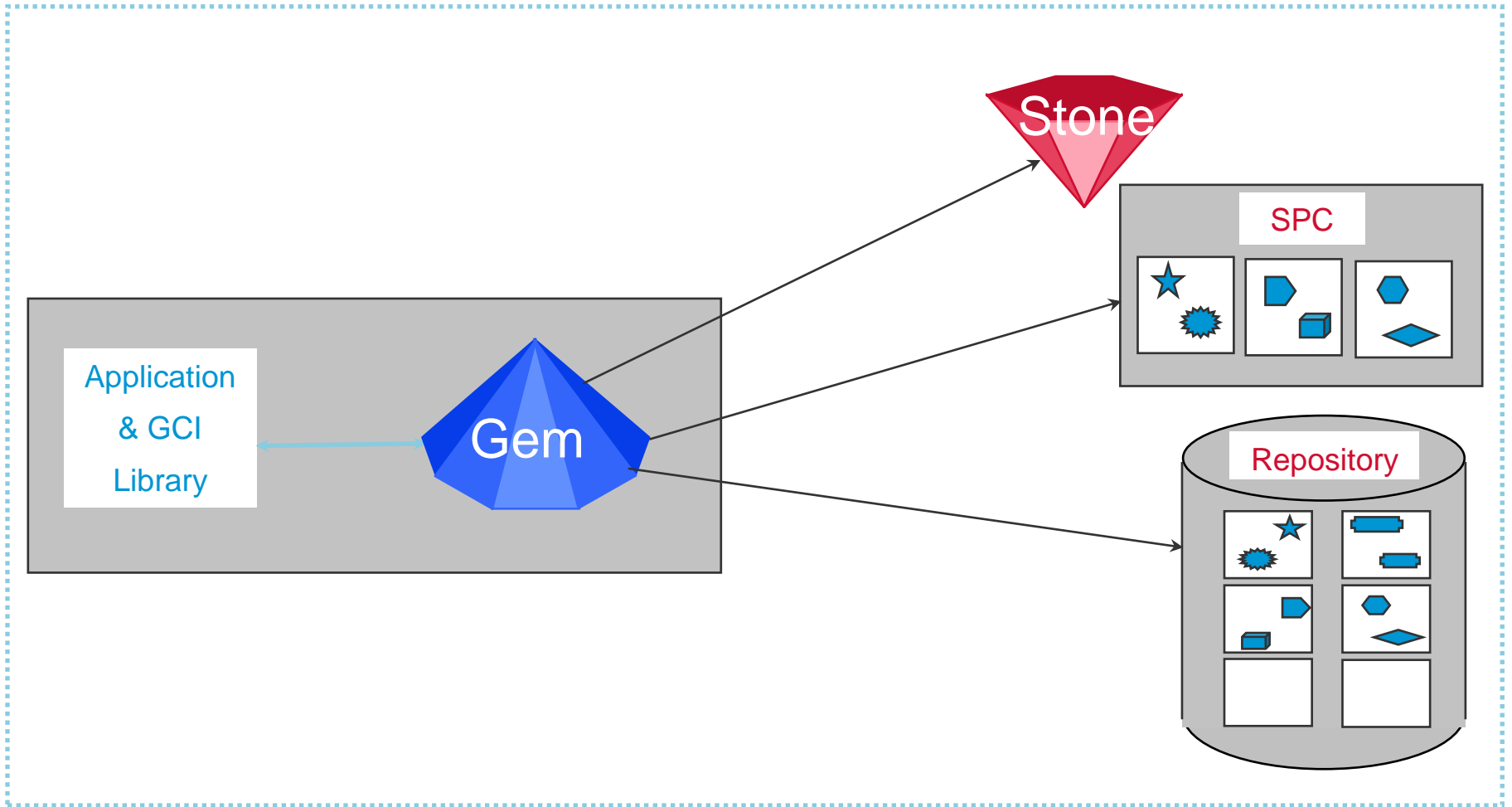
■ Remote Procedure Call (RPC) Gem

- Application loads GCI library into its process space
- GCI library asks NetLDI process to start Gem process
- Gem process can be on same or different host as application
- Additional communications overhead
- Reduced risk of application corrupting Gem



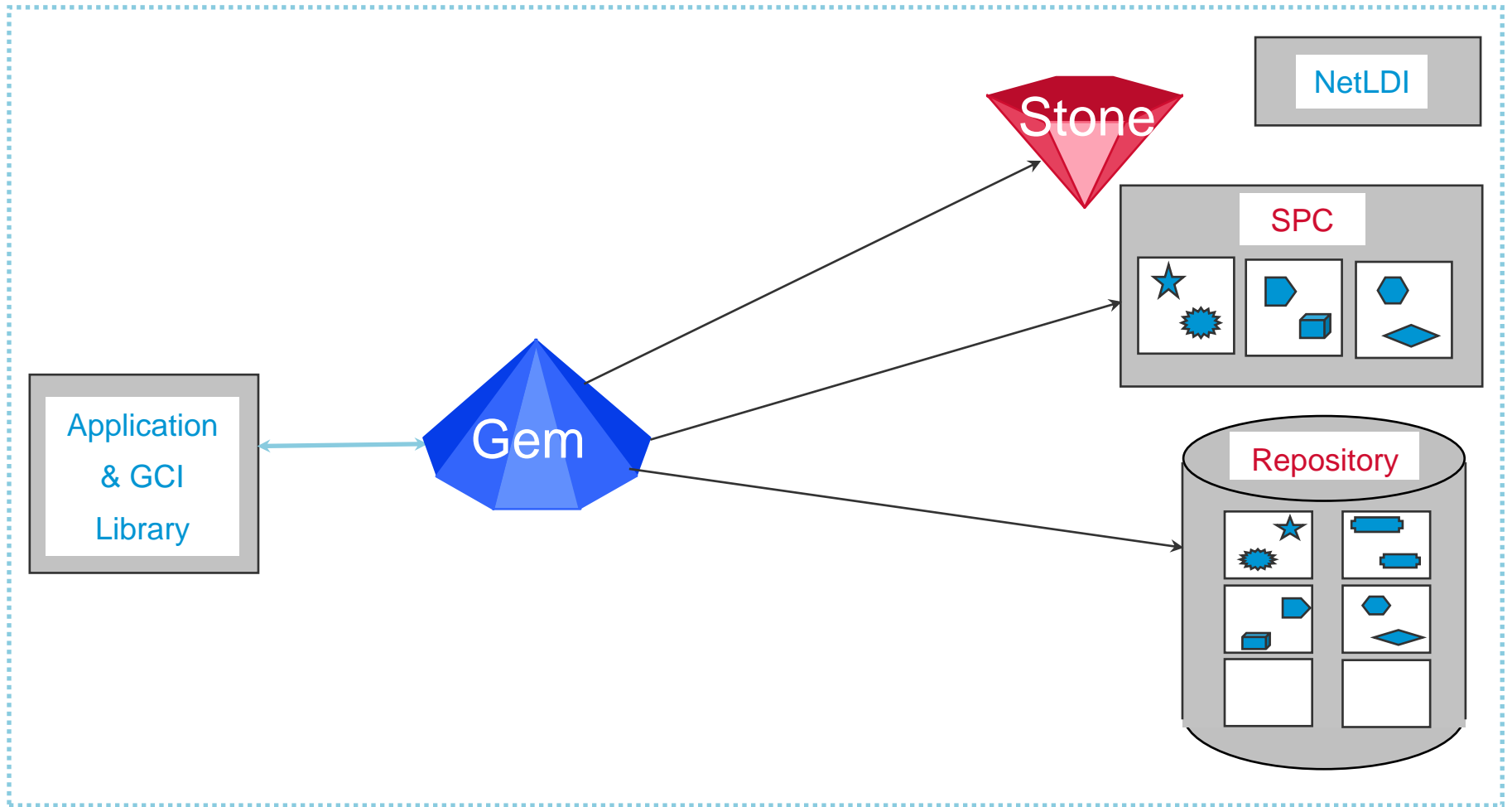
One-Machine Process Locations (Linked Gem)

Stone Host



One-Machine Process Locations (RPC Gem)

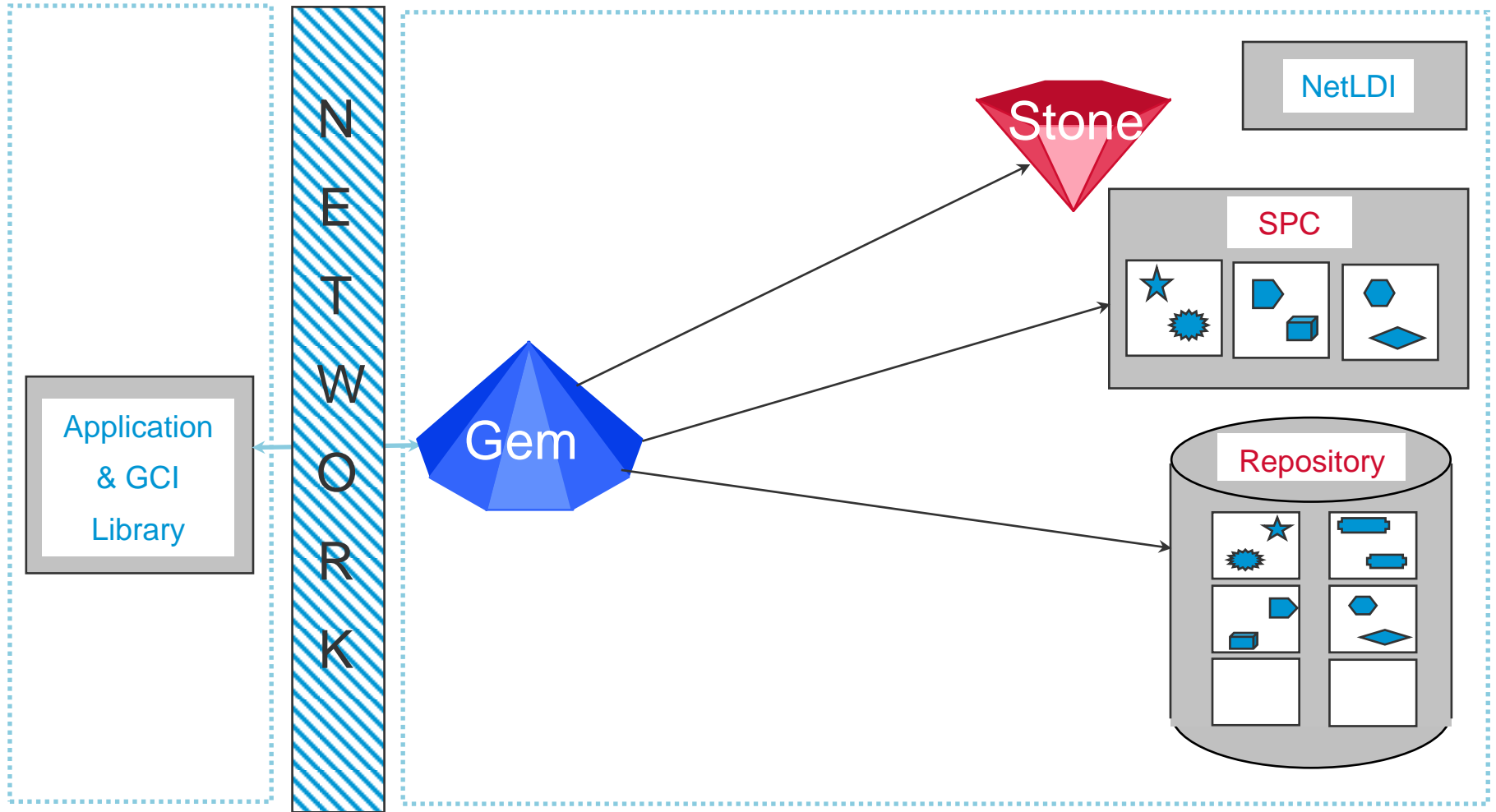
Stone Host



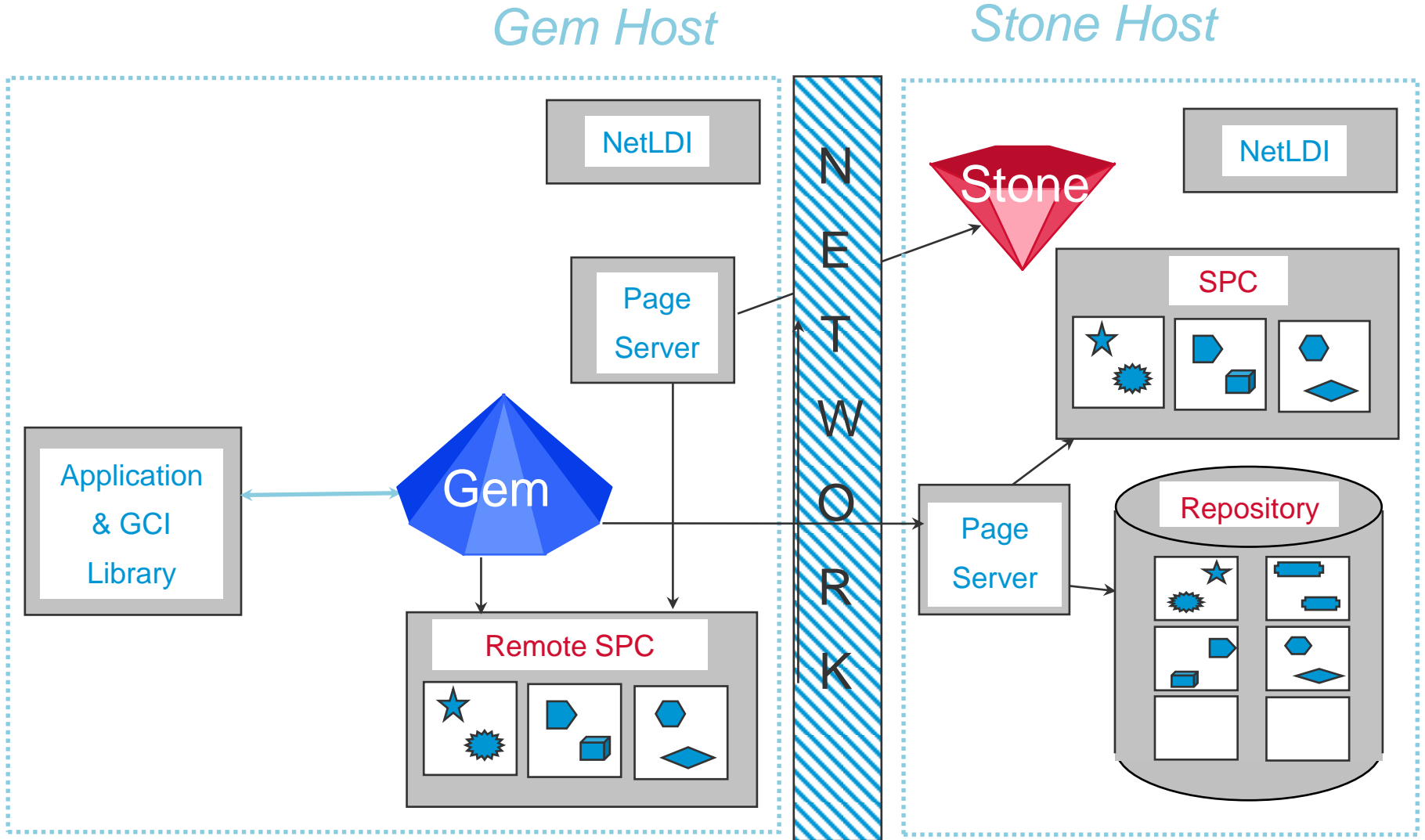
Two-Machine Process Locations (Gem on Stone Host)

Client Host

Stone Host



Two-Machine Process Locations (Gem Remote from Stone)

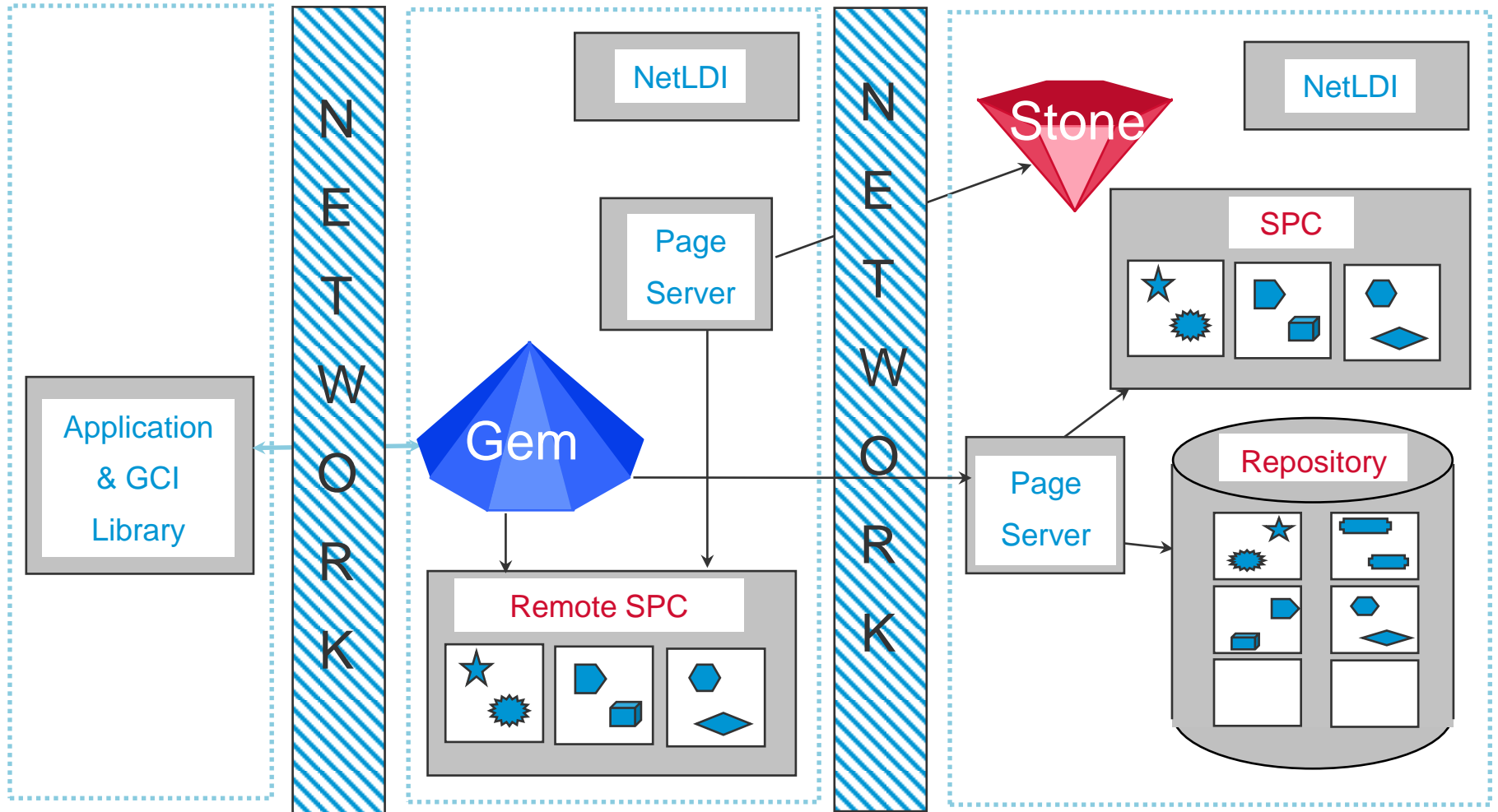


Three-Machine Process Locations

Client Host

Gem Host

Stone Host



Gem Startup

- **Gem process started (if RPC)**
 - Linked Gem started with Application
 - RPC Gem started by NetLDI based on request from Application (via GCI)
- **Application requests login**
 - Application provides Stone host and name to Gem through GCI library
 - Gem contacts Stone and is assigned a session ID and a database view
 - Gem connects to SPC on local machine
 - Stone asks NetLDI on Gem host to start SPC Monitor if needed
 - Application provides user ID and password to Gem through GCI library and Gem validates user based on lookup in database
- **Login complete**
 - Gem now waits for requests from GCI
 - Application submits requests to GCI for Smalltalk execution or objects

Architecture: Database View and Commit Records

Database View and Commit Record

- On login, Gem has a database view

- Object Table**

- Object ID (OID) == Object Oriented Pointer (OOP)
- Map to Page (offset in an Extent)
- Each view is based on a single Object Table

Object Table

Object ID	Page ID
-----------	---------

246	10343
247	10343
248	-1

- Each commit creates a *Commit Record***

- Reference to unique Object Table
- List of modified objects (*Write Set*)
- List of Shadowed Pages

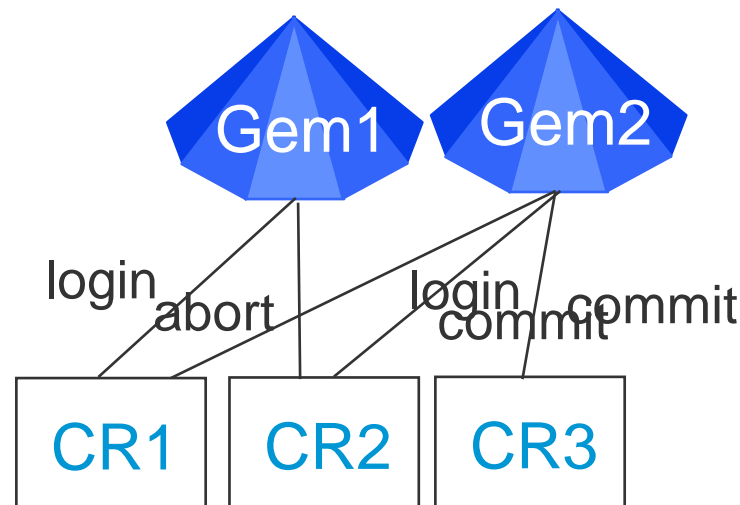
Object Table Reference

Write Set

Shadowed Pages

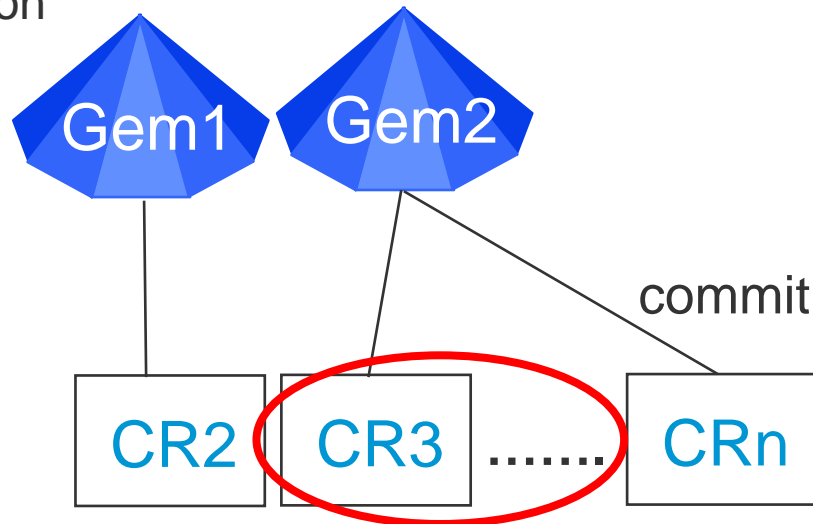
Commit Records

- There is always at least one database view, or Commit Record (CR)
- On login, a Gem is given the most recently created Commit Record
- Other Gems can share the same Commit Record (login or abort)
- Each (non-empty) commit creates a new Commit Record
- An abort moves a Gem to the latest Commit Record
- *Oldest* CR may be disposed of if it is not referenced
- Another commit creates another Commit Record



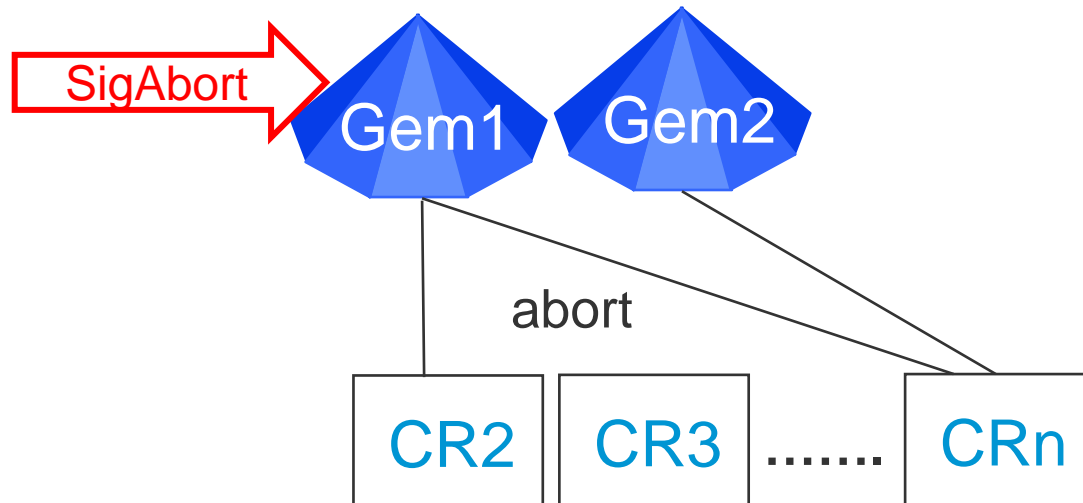
Commit Record Backlog

- Here we have two Gems and two Commit Records
- Additional commits create more Commit Records (maybe many!)
- Intermediate CRs *cannot* be disposed of if older CR is referenced
 - This can be a major performance issue — a large CR Backlog is bad!
- Problems with excess Commit Records
 - They take space in SharedPageCache and/or Repository
 - They slow down new commit processing
 - They delay garbage collection



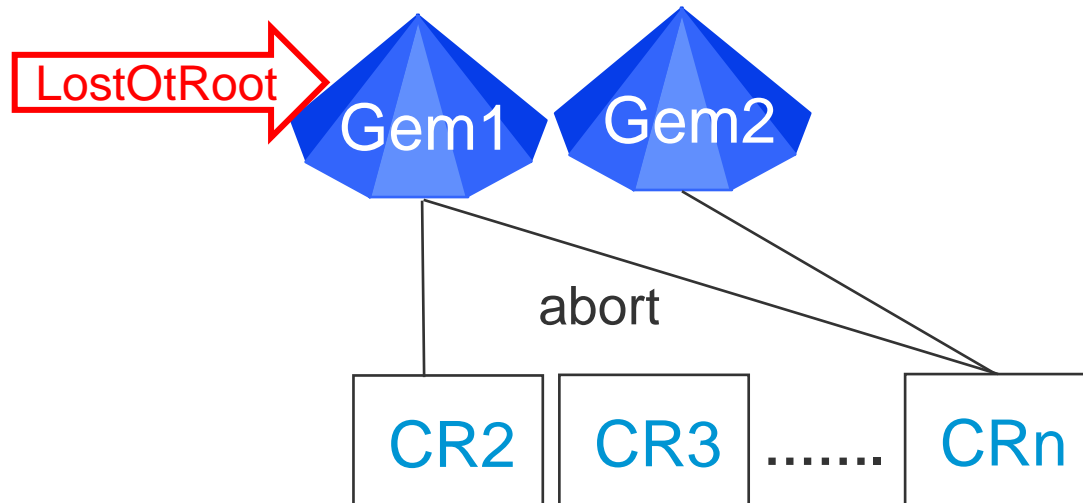
SigAbort

- Important to avoid excessive CR Backlog
- Signal requesting an abort (SigAbort) sent to a Gem if and only if:
 1. Gem is referencing the oldest Commit Record
 2. Gem is not in transaction
 3. CR Backlog is above configured value
- If Gem responds quickly to SigAbort, good!
- Stone can dispose of oldest unreferenced CR(s)



LostOtRoot

- If a **SigAbort** was sent to a **Gem** and it was ignored for **X** minutes
 - X is configurable, with default of one (1)
- Stone will revoke the **Gem's** database view (**Commit Record**)
- Stone will send Gem a signal: **LostOtRoot** (**Lost Object Table Root**)
 - Any object access will give an error
- Stone can dispose of oldest unreferenced **CR(s)**
- Gem must abort to get a new **Commit Record** (or logout)



Transaction State vs. Mode

Transaction *state*

- In – commit attempt is allowed (might succeed or fail)
- Out – commit attempt is not allowed and will always give an error

Transaction *mode*

- #autoBegin – always in a transaction with a stable view
- #manualBegin – can be in or out, but always a stable view
- #transactionless – can be in or out, stable view only when in transaction

	#autoBegin	#manualBegin	#transactionless
Always in	✓		
Stable view in	✓	✓	✓
Stable view out	N/A	✓	
Can get SigAbort		✓	
Can get SigFinish	✓		
Safe for GBS	✓	✓	

Transaction Control

■ **System abortTransaction**

- Abort, losing any existing changes and obtain the most recent Commit Record
- New transaction state:
 - 'In transaction' if transaction mode is #autoBegin
 - 'Out of transaction' if transaction mode is #manualBegin or #transactionless

■ **System beginTransaction**

- Abort, losing any existing changes and obtain the most recent Commit Record
- Enter the 'in transaction' state (for all transaction modes)

■ **System commitTransaction**

- If commit succeeds, new transaction state:
 - 'In transaction' if transaction mode is #autoBegin
 - 'Out of transaction' if transaction mode is #manualBegin or #transactionless
- If commit fails, see next slide ...

Failed Commit

■ Reasons for a commit failure

- Another Gem has a lock (read or write) on an object we modified
- An object we modified was modified in a Commit Record after we got our view

■ Impact of commit failure

- Update to most recent Commit Record
 - No longer on prior Commit Record, so database view is updated; but ...
- Still have all locally modified objects
 - New database view does not change local modifications of persistent objects
- Still in transaction
 - But any further commit attempt will fail
- Gem will need to abort before any subsequent commit can succeed
 - Abort will lose all local modifications of persistent objects
 - May wish to *copy* modifications into other objects before abort
 - Could reapply changes after abort and attempt another commit

Agenda

- Installation
- Architecture
- **Tools**
- Backup/Restore
- Class Versions
- Debugging
- Concurrency
- Background Processing
- Repository-wide Garbage Collection

Tools

Tools

■ Traditional

- GemBuilder for Smalltalk (GBS)
- Topaz
- Visual Statistics Display (VSD)
- Transaction log analysis scripts

■ Built using other Smalltalks

- GemTools (Pharo)
- Jade (Windows/Dolphin)

■ Web

- tODE
- WebTools

GemBuilder for Smalltalk

- **Smalltalk package that wraps GCI library**
 - Cincom Smalltalk (VisualWorks)
 - VA Smalltalk from Instantiations (formerly VisualAge Smalltalk from IBM)
- **Provides "transparent" replication between server Smalltalk and client Smalltalk**
 - Traditionally used to build "rich client" GUI applications
- **Tools**
 - Code browser
 - Debugger
 - Inspector
 - UserProfile editor
 - ...
- **Not available in Web Edition**

Topaz

- **Command-line wrapper for GCI library**
- **Limited scripting capabilities**
 - No control flow (loop/conditionals)
- **Used extensively by GemStone's internal developers**
 - Used to load code (see \$GEMSTONE/upgrade)
 - Used to test server
 - Preferred way to report server product bugs
- **Primary customer use is for batch jobs**
 - SystemRepository>>#fullBackupTo:
 - SystemRepository>>#markForCollection
- **Useful when GUI-based tools are not available**
 - SSH to production server behind firewall over WAN

Visual Statistics Display (VSD)

- **Each process records data to shared page cache (SPC)**
 - Many values exist and are constantly updated (220 for Gems, 272 for Stone)
 - Any process attached to SPC can monitor other processes on same host
- **\$GEMSTONE/bin/statmonitor**
 - Periodically copies current data to a file for later analysis
 - Every production system should be capturing statistics for later analysis
 - If Gems are on separate machine from Stone, need additional statmonitors
- **VSD application**
 - \$GEMSTONE/bin/vsd contains an X Window System application (Linux/Mac)
 - Window version at <http://community.gemstone.com/display/GSS64/VSD>
- **Vital uses**
 - Performance tuning
 - Crash analysis

Transaction Log Analysis Scripts

- **All changes to persistent objects take place in a transaction**
 - Record of each transaction in transaction logs
- **Primary use is to replay transactions**
 - After restore from backup
 - Restart after crash
- **Scripts are available to search tranlogs**
 - Traditional use is for bug analysis ("How did that happen!?")
 - Recent enhancements for forensic analysis ("Who made that change?")
- **System Administration Guide, Appendix H**
 - `$GEMSTONE/bin/printlogs.sh`
 - `$GEMSTONE/bin/searchlogs.sh`

GemTools

■ **Pharo-based GUI application**

- Uses Squeak's FFI interface to interact with GCI library
- Pharo is used to provide GUI (primarily with OmniBrowser)
- No support for object replication—pretend that Pharo Smalltalk doesn't exist

■ **Tools**

- Code browser, Monticello browser, Metacello browser
- Workspace, Debugger, Inspector
- Backup/restore menus

■ **Closely tied to GLASS**

- Use of OB means many round-trips to server for each action

■ **Download from <http://seaside.gemstone.com/downloads.html>**

Jade

- **Windows-only stand-alone 1 MB executable build with Dolphin**
 - Works with all GemStone/S versions (32-bit and 64-bit)
 - Does not require any server code to be pre-loaded
 - Optimized for slow network (most operations require only one round-trip)
- **Tools**
 - Code browser, Monticello browser
 - Workspace, Debugger, Inspector
- **Download from <http://seaside.gemstone.com/jade/>**

tODE – the Object (Centric) Development Environment

■ Seaside-based web application

- Runs in Pharo and GemStone
- Non-traditional approach to tools

■ Tools

- Code browser, Metacello browser
- Workspace, Debugger, Inspector

■ Web Resources

- Code at <http://code.google.com/p/tode/>
- Mailing list at http://groups.google.com/group/tode_st

WebTools

- **Javascript application**

- Uses async Json queries to lightweight web server in GemStone
- Not a Seaside application; available in any GemStone/S 64 Bit 3.0 database

- **Tools**

- Code browser
- Statmonitor file graphing

- **Extensible with plug-in tools**

- **Code distributed in `$GEMSTONE/examples/www/`**

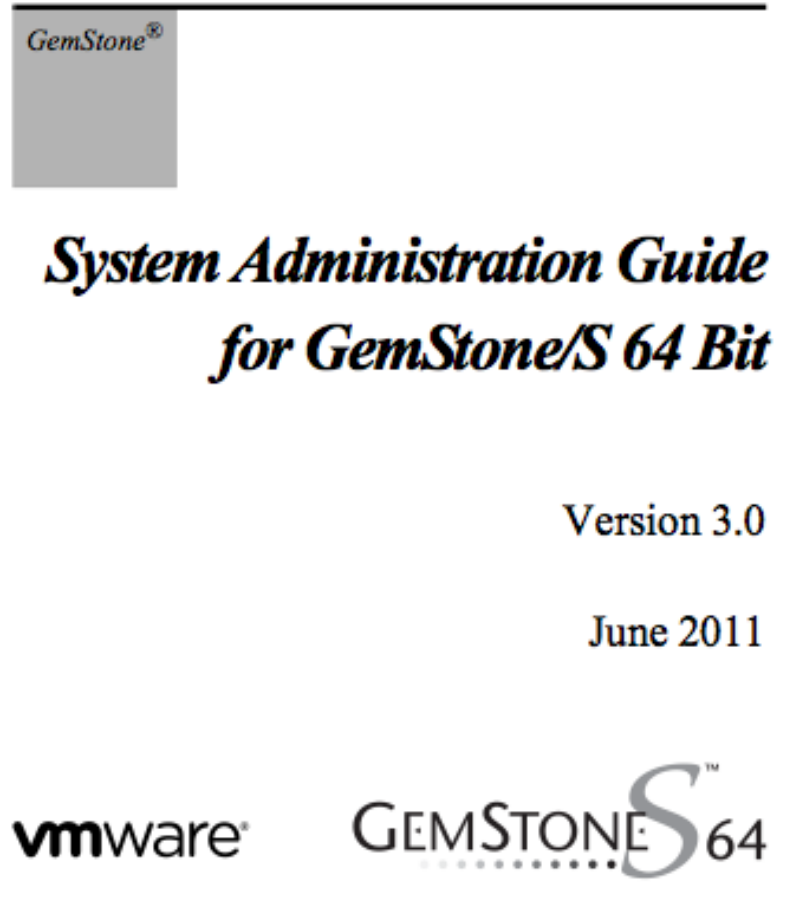
Agenda

- Installation
- Architecture
- Tools
- **Backup/Restore**
- Class Versions
- Debugging
- Concurrency
- Background Processing
- Repository-wide Garbage Collection

Backup and Restore

Documentation

- System Administration Guide for GemStone/S 64 Bit
- Chapter 9: Making and Restoring Backups



Types of Backups

- **Off-line Extent Copy**
- **Smalltalk Backup**
- **On-line Extent Copy**

Off-line Extent Copy

- **Make a copy of extent(s) when system is down.**
- **Advantages**
 - Simple to make
 - Simple to restore
- **Disadvantages**
 - System must be down
 - Copy includes empty space and non-object data
- **Recommended for**
 - Development systems
 - Systems that are regularly down due to usage patterns and hardware needs

Smalltalk Backup

- **Evaluate 'SystemRepository fullBackupTo: aFilePath'**
 - Alternative method: #'fullBackupCompressedTo:'
 - Other methods support multi-file backups to limit file size (for tape backups!)
- **Advantages**
 - Traditional means
 - Compact size contains only objects
 - Restore results in minimal extent size(s)
- **Disadvantages**
 - Can take significant time to create backup
 - Gem process holds a transaction for initial phase
 - Restore process more time-consuming
- **Recommended for:**
 - Smaller systems that have not experienced any of the disadvantages

Hybrid: On-line Extent Copy

■ Process

1. Suspend checkpoints
2. Copy extent(s)
3. Resume checkpoints

■ Advantages

- On-line
- Most work is done by OS file copy (as compared to Smalltalk backup)
- Can take advantage of hardware features (split a mirror)

■ Disadvantages

- More complex to create & restore
- Backup might be invalid if checkpoints resumed too early

■ Recommended for:

- Larger systems

Recomendation

- **Validate backup file**
 - `copydbf extentOrBackup /dev/null`
- **Test your restore process**
- **Keep transaction logs associated with backup**

Agenda

- Installation
- Architecture
- Tools
- Backup/Restore
- **Class Versions**
- Debugging
- Concurrency
- Background Processing
- Repository-wide Garbage Collection

Class Versions

- GemStone/S 64 Bit Programming Guide
- Chapter 9
 - Class Creation,
 - Versions,
 - and Instance Migration

The logo consists of the word "GemStone" in a serif font with a registered trademark symbol, positioned above a solid grey rectangular box.

GemStone/S 64 Bit Programming Guide

Version 3.0

June 2011

vmware®

GEMSTONE[™]
.....S 64

Class Creation

- **Send `#'subclass:....'` message to object that will be superclass**
- **What happens in traditional Smalltalks if class already exists?**
 - Create new class object
 - Copy and recompile methods to new class
 - Find all instances of old class
 - Create new instances of new class for each instance of old class
 - Copy values in instance variables from old to new instances
 - Perform `#'become:'`-like action to swap old and new instances
 - Perform `#'become:'`-like action to swap old and new classes
 - Garbage collect to remove old class and instances
- **Can't do this in GemStone**
 - Scanning large object space would take too long
 - Modifying objects in other session's views would violate isolation

Class Versions

- **Every class is part of a ClassHistory collection**
 - Others are related but may have different name and/or schema
 - Might be only one Class in ClassHistory
- **Object>>#'isKindOf:' checks superclasses *and* ClassHistory**
- **Most compiled references to a Class will be updated automatically**
 - Methods do not reference a Class, but a SymbolAssociation with a value
- **Methods are typically copied and recompiled by tools**
 - Low-level subclass creation does not automatically create methods
- **Instances are left with old class until migrated explicitly**

Instance Migration

- **Could leave instances of old class as-is**
 - Provide method(s) that answer default values for missing instance variables
- **Could migrate all at once**
 - Make instance migration part of general application upgrade process
 - Application down-time
 - Find all instances and migrate them in one or more transactions
- **Could do lazy migration**
 - Modify code to migrate before any message is sent to old object
 - Simple if limited lookup path(s) to object
 - Elaborate approach of replacing methods on old class to migrate self
 - Could still have background process to finish migration as soon as possible

Agenda

- Installation
- Architecture
- Tools
- Backup/Restore
- Class Versions
- **Debugging**
- Concurrency
- Background Processing
- Repository-wide Garbage Collection

Debugging

Debugging – Printing/Logging

- **System class>>#'addAllToStoneLog:'**
- GsFile class
 - #'stdout'
 - #'stderr'
 - #'gciLogClient:'
 - #'gciLogServer:'
- **TranscriptProxy class>>#'show:'**
 - Global Transcript points to TranscriptProxy class
 - Creates an ObjectLogEntry
 - Will send to client if client has registered a ClientForwarder
 - Otherwise sends GsFile class>>#'gciLogServer:'
- Store value in global
 - UserGlobals at: #'James' put: 'got to step #1 at ' , DateTime now printString

Debugging – Halt and Breakpoints

- **Object>>#'halt'**
 - Signals a Halt exception
 - Might be trapped by Exception handlers
 - Typically reports exception back to GCI client (Topaz, GemTools, etc)
 - Similar behavior for most other Error exceptions
- **Set breakpoint in method**
 - Does not require modifying source code
 - Applies only to current Gem
 - Require tool support (or GsNMethod>>#'setBreakAtStepPoint:')

Remote Debugging

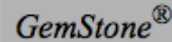
- **DebuggerLogEntry (subclass of ObjectLogEntry)**
 - Application may persist a continuation with an error
 - DebuggerLogEntry class>>#'createContinuationLabeled:'
 - Tools may support opening a debugger on persisted continuation

Agenda

- Installation
- Architecture
- Tools
- Backup/Restore
- Class Versions
- Debugging
- **Concurrency**
- Background Processing
- Repository-wide Garbage Collection

Concurrency

- GemStone/S 64 Bit Programming Guide
- Chapter 7
 - Transactions and Concurrency Control

The logo consists of the word "GemStone" in a serif font with a registered trademark symbol, positioned above a solid grey rectangular box.

GemStone/S 64 Bit Programming Guide

Version 3.0

June 2011

vmware[®]

GEMSTONE[™]
.....S64

Concurrency Issues

- **GemStone prevents simultaneous updates to same object**
 - Each session starts with a database view
 - First to commit wins
 - Other sessions will get a TransactionConflict error if objects have changed
 - This is "optimistic locking"
- **Short transactions reduce likelihood of this "physical" conflict**
 - Abort just before making change and then commit immediately
 - Note that value might change based on abort!

Logical Conflict

- **Seaside framework (mostly) addresses physical conflict problem**
 - Abort performed immediately before executing callbacks and rendering page
 - TransactionConflict error is handled by abort then retrying (up to 10 times)
 - Odds are that another attempt will succeed
- **Application is responsible for detecting "logical" conflicts**
 - User may enter data based on old view of database
 - Abort may switch to newer view with different data than presented to user
 - Seaside will do abort and then replace existing (possibly changed) value with user's entry

Explicit Locking

■ Pessimistic Locking

- System class >> #'writeLock:' (and friends)
- If you are successful in obtaining a write lock, then no other session may commit a change to that object
- Your view might be out-of-date, however, and you need an abort/commit before modifying the locked object

Reduced Conflict Classes

- **Certain overlapping modifications to an object might be okay**
 - Multiple sessions adding objects to a collection
 - Adding, changing, or removing the value at different keys in a Dictionary
 - Incrementing a counter
- **GemStone provides classes that avoid well-defined conflicts**
 - RcCounter
 - RcIdentityBag
 - RcQueue
 - RcKeyValueDictionary
- **Trade-off of slight overhead for avoiding conflicts**

Agenda

- Installation
- Architecture
- Tools
- Backup/Restore
- Class Versions
- Debugging
- Concurrency
- **Background Processing**
- Repository-wide Garbage Collection

Background Processing

Session View

- **A single VM (Gem) has a single database view**
 - A forked Process (via ExecutableBlock>>#'fork') runs in the same view
 - A commit or abort by any Smalltalk Process in the session will change the view for all Smalltalk code running in the Gem
- **Typical Smalltalk patterns will not work**
 - Fork a Process to handle a web request using a unique ODBC connection
 - Fork a Process to handle a long-running background task using a unique DB
 - Fork a Process to handle web requests and edit code in foreground
- **Need a separate Gem for each independent activity**

Background Processing In GemStone

- **Define a 'cron' job for regular maintenance (backup and MFC)**
- **Start a dedicated Topaz session for background jobs**
 - Multiple "producers" add tasks to a well-known collection (e.g., an RcQueue)
 - Single "consumer" takes tasks from queue, and processes them
 - Gem does only one task at a time, in a transaction
 - On TransactionConflict error, abort and start over

Agenda

- Installation
- Architecture
- Tools
- Backup/Restore
- Class Versions
- Debugging
- Concurrency
- Background Processing
- **Repository-wide Garbage Collection**

Repository Garbage Collection

Types of Repository Garbage Collection

■ Page reclamation

- Recovering space taken by shadow objects
- Compacting space by copying objects from partially filled pages

■ Full markForCollection

- Scan the entire repository for any references to every object

■ Epoch GC

- Scan the objects modified during a time period (epoch) for new references to new objects

■ Off-line GC

- Scan the entire repository for any references to objects found to be unreferenced by an off-line scan

Garbage Collection Vocabulary

■ AllUsers

- The instance of UserProfileSet that acts as a root for the object graph

■ Live object

- An object referenced directly or indirectly from AllUsers

■ Dead object

- An object defined in the object table and present on a page, but not live
- A dead object may be referenced by a dead object (but not a live object)
- A dead object may reference both live and dead objects (but it doesn't matter)
- The object ID (OOP) and the space of a dead object may be reclaimed

■ Shadow object

- When an existing object is modified, it is placed on a new page
- The old page is preserved until no more views reference the old object
- The space can be reclaimed (through page compaction), but not the object ID

Summary of Repository-Wide Garbage Collection

- **MFC Gem builds possible dead set**
 - Mark live objects
 - Object table sweep
 - Record possible dead
- **Voting to remove from possible dead set (managed by Stone)**
 - Current gems vote based on current references at next commit or abort
 - GcGem votes on behalf of all commit records since start of MFC
- **Cleanup**
 - Finalizing for selected objects
 - Page reclamation
 - Return of pages and object IDs to free pool

Mark Live Objects

■ Find connected objects

- Start with AllUsers as the root of the object graph (the original 'live' object)
- Perform a 'transitive closure' visiting each object referenced from a live object
- Add each live object to a live object set

■ Gem: ProgressCount

- Number of live objects found so far
- When this statistic drops back to zero, this step is done

■ Configuration

- Set mfcGcPageBufSize

■ Process is very I/O and CPU intensive

- Read object table page and data page for every live object
- Same page might be read multiple times

Object Table Sweep

- Subtract *live* objects from *all* objects to get possible dead set
- **Gem: ProgressCount**
 - Begins at zero (clearing from previous step)
 - Count of possible dead objects
 - When this statistic drops back to zero, this step is done

Record Possible Dead

- Pass possible dead set to stone
- MFC Gem's task is now done
- **Stn: PossibleDeadSize**
 - Rough approximation of possible dead set size

Voting by Existing Gems

- **As each logged-in Gem does an abort or commit**
 - Stone passes list of possible dead to Gem for voting
 - Gem scans its private memory for references to the objects
 - Referenced objects are voted 'not dead'
- **Gem Statistic**
 - VoteNotDead
- **Stn Statistics**
 - GcPossibleDeadSize
 - GcVoteUnderway
 - SessionNotVoted
- **Garbage collection can stall here**
 - Voting happens only at the next abort or commit
 - A quiet Gem that does not abort or commit will not vote

Voting by GcGem ('Finalize Voting')

- Original live object set is based on view at beginning of MFC
- Any commit since MFC began could have created a reference
- A 'write set union' of all commit records since MFC began is kept
- GcGem takes possible dead set and searches for new references
- Stn statistics:
 - GcPossibleDeadWsUnionSize
 - GcSweepCount
 - GcPossibleDeadSize
 - DeadNotReclaimedSize
- Gem statistic:
 - ProgressCount
- At end, we have a definitive dead object set

Cleanup: Finalizing

- **GcGem reads each object in the dead set**
- **Certain dead objects require special cleanup**
 - Collections with indexes
 - Compiled methods
- **Gem: ProgressCount**
- **Stn: GcPossibleDeadSize**
- **Stn: DeadNotReclaimed**

Cleanup: Reclamation

- **GcGem activity**
- **For each page containing a dead object**
 - In a transaction, copy all live objects on that page to a new page
 - This leaves only shadow objects (the current version is on a new page), dead objects, and free space on the old page
- **Note that the old page might still be referenced from a view**
 - Shadow objects need to be kept around as long as they are part of a view
- **Stn statistics**
 - GcReclaimState
 - GcReclaimNewDataPagesCount
 - DeadObjsCount
 - FreePages
 - GcPagesNeedReclaimSize
 - DeadNotReclaimedSize

Cleanup: Return to Free Pool

- **When commit record for reclaim activity is no longer referenced**
- **Page IDs and Object IDs associated with that reclaim are added to the free list**

Questions?

- **James Foster**

- jfoster@vmware.com
- <http://programminggems.wordpress.com/>