



Product Overview

Norman R. Green
norm.green@gemstone.com
Director of Engineering
GemStone Systems Inc.

Subjects For Today's Talk



- GemStone Systems At A Glance (briefly)
- What Is GemStone/S? (briefly)
- Why 64 Bit?
- Phase 1
- Phase 2
- Smalltalk Coding Changes
- Future Features



GemStone Systems at a Glance

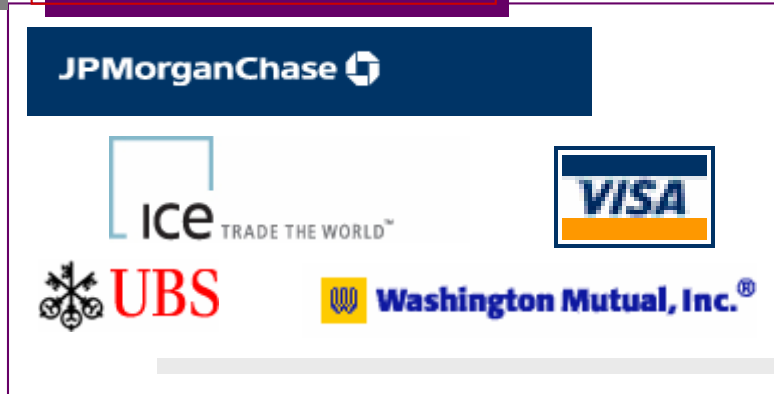


- Founded 1982
- Headquarters: Beaverton, Oregon, USA
- Privately Held
 - Investor: 65%
 - Employees: 35%
- Experienced Engineering Team
 - Many employees with 10+ years tenure
 - Some over 20 years.
- Over 200 installed customers
- 24 x 7 global support.

GemStone/S Customers



Banking / Finance



Government



Telecommunications



Transportation/Shipping



Three Easy Pieces:

1. An Object Oriented Database

- **Objects stored in object format**
 - **No tables, rows, or columns.**
 - **No O/R mapping code to maintain.**
- **ACID Transactions**
 - **Atomic – all or nothing**
 - **Consistent – start/end states are consistent**
 - **Isolation – Commit in Session A does not immediately affect Session B**
 - **Durable – cannot be rolled back once committed.**



2. A Smalltalk dialect

- **Comes with a complete set of Kernel Classes.**
- **Repository contains object behavior (classes, compiled methods, etc) as well as object state (inst vars, class vars, etc).**

3. An Application Server

- Applications have been written entirely in GemStone/S.
- Supports many kinds of clients
 - Smalltalk: VA & VW
 - Java
 - C & C++
 - UNIX shell (using topaz)

- **Customers desired to scale beyond the 32-bit limits:**
 - **4 GB Addressable Memory**
 - **Max shared page cache size = 2 - 3 GB**
 - **1 billion objects max**

- **Why Do We Care About 32 Bit Limits?**
- **Examples**
 - **5 GB repository, 2 GB Shared page cache**
 - 40% of database cached
 - Performance: **OK**A yellow emoji with a wide smile and blue eyes, giving a thumbs-up gesture with its right hand.
 - **100 GB repository, 2 GB Shared page cache**
 - 2% of database cached
 - Performance: **POOR**A yellow emoji with a sad expression, closed eyes, and blue tears streaming down its face.

- **Project Divided Into 2 Phases:**
 - **Phase 1: 64 bit address space**
 - **Phase 2: 64 bit object identifiers**



Phase 1 Goals

- Performance Scalability
 - Exploit 64 bit address space
 - Improve Smalltalk VM performance
 - Reduce persistent garbage

Phase 1 Features

- 100% 64 bit
- 2 billion OOP limit
- Current Version: 1.1.5
- Supported Platforms :
 - Solaris 9, 10
 - HPUX 11.11 (PA-RISC)
 - AIX 5.2, 5.3
 - Linux x86_64 (RH 4, SuSE 9.3)

Phase 1 Features

- GemBuilder for Smalltalk (GBS) Support
 - VisualWorks 5i.4, 7.4+
 - Visual Age 6.0, 7.0

Phase 1 Improvements

- Supports very large Shared Page Caches
 - Up to 16 TB
 - “Database-in-memory” capable

Phase 1 Improvements

- Shared Cache Warming
 - Loads data into the shared cache
 - Uses any number of gem processes
 - Stops when the shared cache becomes full or all data was loaded.

Phase 1 Improvements

- No more “large object leaks”
 - Old Design
 - Large objects (> 8 Kb) always go to disk
 - Old space overflows go to disk
 - Result: too much persistent garbage
 - New Design
 - Objects only go to disk if referenced by a committed object.
 - No exceptions for large objects.
 - Net effect: Large reduction in garbage object creation.

Phase 1 Improvements

- Improved Symbol Management
 - Symbol Gem
 - New session dedicated to managing AllSymbols.
 - Creates all new symbol objects.
 - Guarantees symbols are always canonical
 - Faster Symbol Lookup
 - AllSymbols collection redesigned for speed
 - Lookups now use binary searches

Phase 1 Improvements

➤ Online Backups

- Safely copy database extents while system is running.
- No impact to users: commits and aborts are allowed.

Phase 1 Improvements

- Parallelized Garbage Collection
 - Reclaim GC gems
 - Variable number: 0 to 255
 - Reclaim both shadow and dead objects
 - Run “online” with production
 - Improved reclaim performance for large production systems.

Phase 1 Improvements

- Major Virtual Machine Redesign
 - Copy-on-read Object Manager (OM)
 - All Classes and Super classes read into private memory.
 - All objects read/written are copied to private VM memory.
 - Large working set = large memory footprint
 - Byte code dispatch loop written in assembler

Phase 1 Improvements

Faster Smalltalk Virtual Machine

Test	GS 6.1	GS64 1.1
50 factorial	2.459	2.175
100 factorial	2.397	2.092
Commit 6.5 MB data	54.541	20.136
Fault and verify 30 MB data	3.478	2.469
Create & de-ref 6.5MB objs	3.8879	0.3809

Phase 1 Engineering Tasks

- For 615,000 lines of C code:
 - Convert to C++
 - Change of compilers
 - Make 64-bit safe
 - long -> int
 - unsigned long -> unsigned int
 - Rewrite object manager and LOM garbage collector from scratch
- Duration: 15 months

Phase 1 Production Customers

- China Ocean Shipping Company (COSCO)
 - Location: Shanghai, China
 - Business: Container Shipping
 - Platforms: HPUX, Solaris, Windows XP
 - URL: www.cosco.com
- Intercontinental Exchange (ICE)
 - Location: Atlanta, GA
 - Business: Energy Futures Trading
 - Platforms: AIX, Windows XP
 - URL: www.theice.com

Phase 1 Production Customers

- Northwater Capital
 - Location: Toronto, Canada
 - Business: Financial Portfolio Management
 - Platforms: Linux, Solaris
 - URL: www.northwatercapital.com
- Soon: Dutch Agricultural Institute (LEI)
 - Location: The Hague, Netherlands
 - Business: Government
 - Platforms: Solaris, Windows
 - URL: www.lei.nl

Phase 2 Goals

- Object Volume Scalability
 - Support very large repositories
 - Overcome 2 billion object limit
 - Fully exploit 64 bit object identifiers

Phase 2 Features

- 64 bit object IDs
 - Maximum Objects: 2^{40} objects
- Support for Very Large Databases
 - Max # of database extents: 255
 - Max extent size: 32 TB
 - Max database size: 8,160 TB

Phase 2 Features

- Faster Smalltalk VM
 - Add 100 Additional Smalltalk byte codes
 - VM runs 30 – 50% faster
- Extended SmallInteger Range
 - $\pm 2^{60}$ ($\pm 1,152,921,504,606,846,976$)
 - Previous range: $\pm 2^{29}$

Phase 2 Features

- New Special Class: *SmallDouble*
 - Subset of IEEE 754 float format
 - 8 bit exponent
 - 52 bit mantissa
 - Range: $5.0e-39$ to $6.0e+38$ (approx)
 - Specials encapsulate their value in the in the object ID
 - No disk I/O

Phase 2 Features

- Better Indexes on Collections
 - Complete redesign of the indexing subsystem.
 - Faster index creation/removal
 - Fewer concurrency conflicts
 - Same query semantics as before

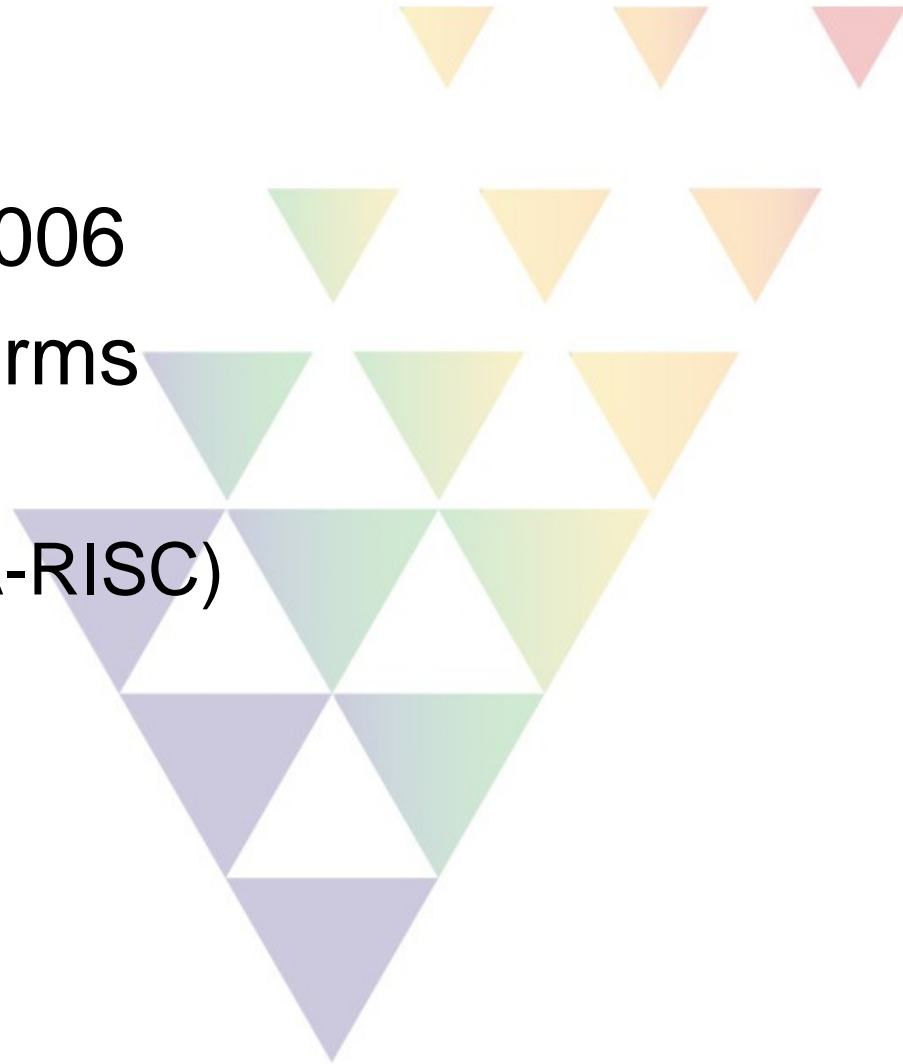
AllEmployees detect:{:e| e.firstName = 'Stephane' & e.lastName = 'Ducasse' }

GemStone/64 2.x Features

- Conversion From Previous Releases
 - GemStone/64 1.1 or later
 - GemStone/S 6.1.5 or later.

GemStone/64 2.0

- Released 3/31/2006
- Supported Platforms
 - Solaris 9, 10
 - HPUX 11.11 (PA-RISC)



Production Customers

- OOCL (Orient Overseas Container Limited)
 - Location: Hong Kong, San Jose, CA
 - Business: Container Shipping
 - Platforms: Solaris, HP-UX, Windows XP
 - URL: www.oocl.com

Production Customers

- OOCL
 - Production deployment: 7/30/06
 - 360 GB repository
 - 1.7 billion objects
 - 1800 concurrent users

OOCL

- Performance:

- Server VM:

- 30% – 50% faster
 - New Smalltalk byte codes


- Client – Server Network load:

- 20% more bytes
 - 32 -> 64 bit oops

OOCL

- Performance:
 - VW Client
 - 30% – 50% Slower
 - 64 bit object IDs are LargePositiveIntegers
 - GemBuilder For Smalltalk (GBS) performance loss
 - Improved GBS coming in October
 - Overall Application Performance: 15% slower.

GemStone/64 2.1

- Target: October 2006:
 - Add support for:
 - AIX 5.2, 5.3
 - Linux on x86-64
 - VM Performance Improvements
 - Index improvements
 - Bug fixes
- 
- A decorative graphic consisting of a large, downward-pointing triangle formed by several smaller, overlapping triangles. The colors of these triangles include shades of purple, green, yellow, and orange.

Possible Problems

- In the new GS64 VM design:
 - All objects accessed by the interpreter must be read into the VM, even if the object is only referenced by identity.
- This was not true before GemStone64

Possible Problems: Example

So this code...

```
aLargeArray do[:each| each == someObject  
  ifTrue:[^each]. ].  
^Object error: #keyNotFound.
```

...may be fast in GS 6.1, but not in GS 64.

Who Knows Why ?

```
aLargeArray do:[ :each | each == someObject  
  ifTrue:[ ^each ]. ]  
^Object error: #keyNotFound.
```

GS 6.1:

- == compares identity without faulting each.

GS 64:

- == compares identity, but each must be read into memory.

Solution: Do it this way:

```
|index|  
index := aLargeArray indexOfIdentical: someObject.  
index == 0  
    ifTrue:[Object error: #keyNotFound]  
    ifFalse:[^aLargeArray at: index].
```

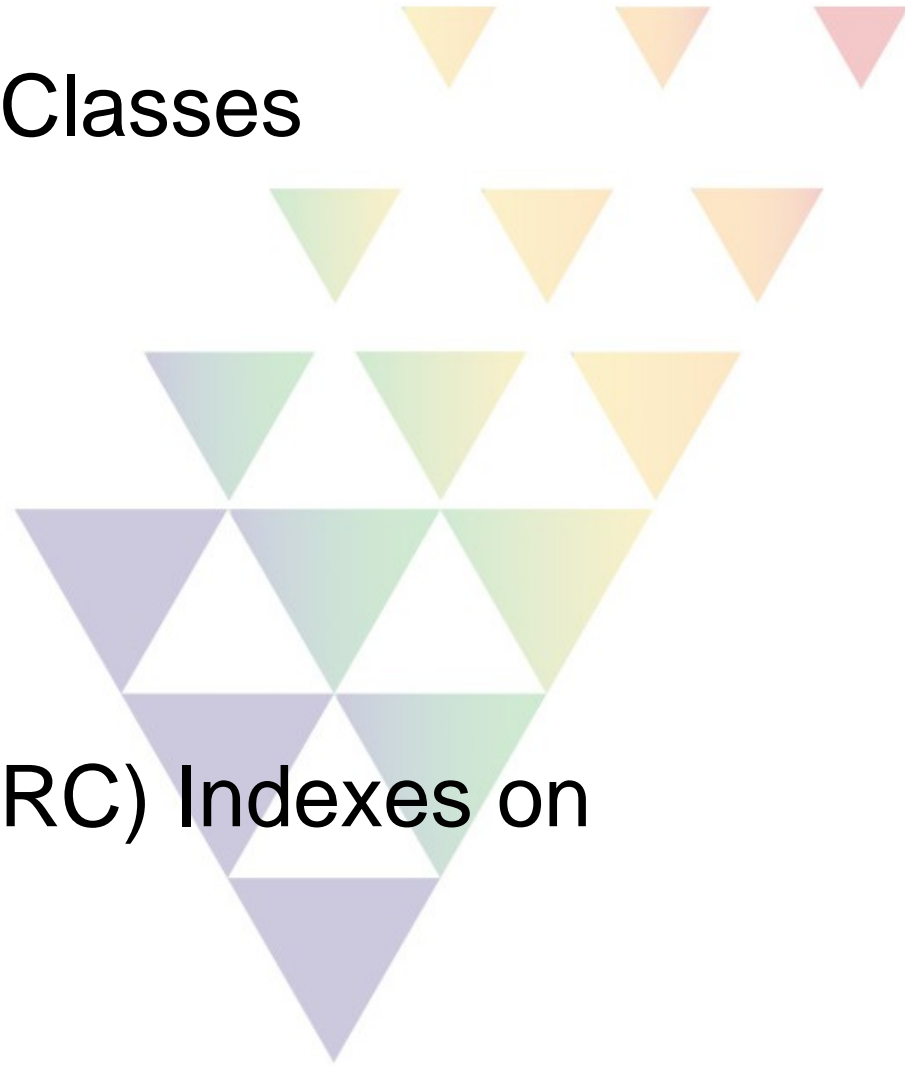
`#indexOfIdentical` **calls a primitive,**
which is smart enough to search
without faulting in the objects in
`aLargeArray`

Other Methods Added To Avoid Unnecessary Object Faulting

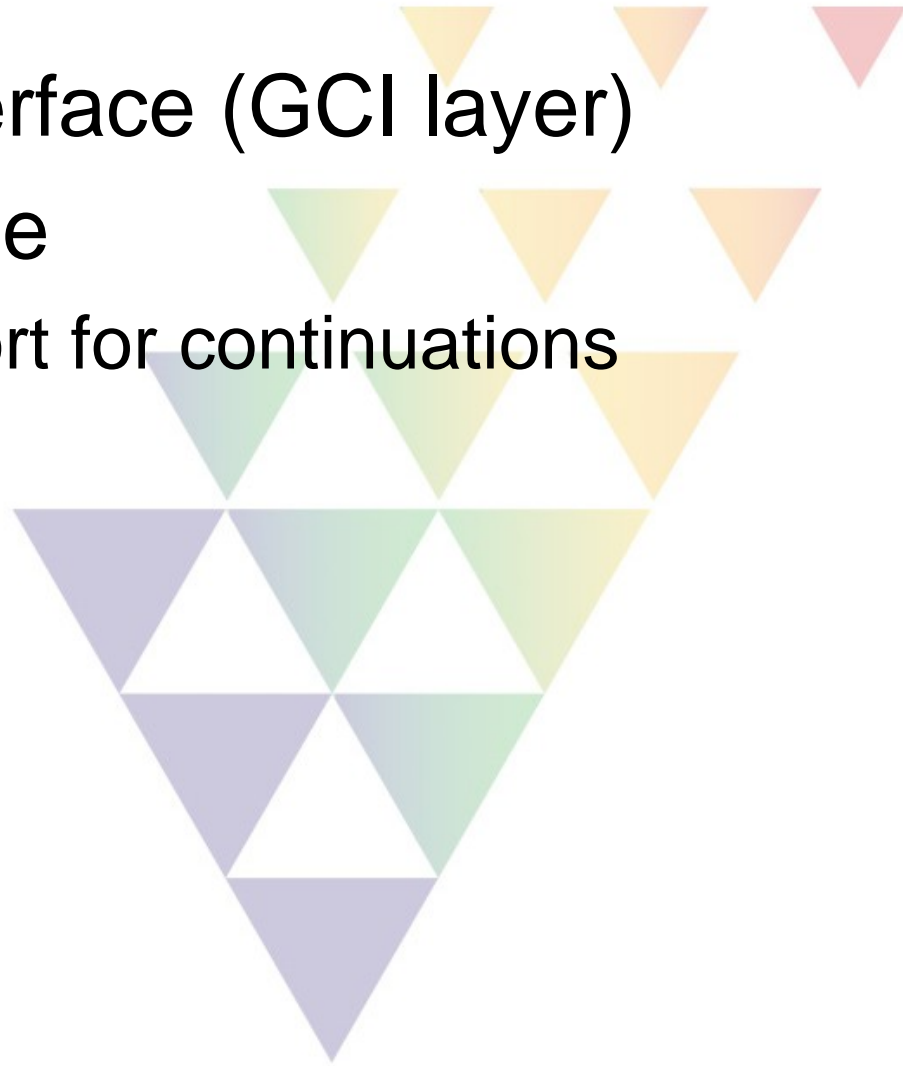
- `IdentityBag >> copyFrom:count:into:startingAt:`
- `IdentityBag >> copyFrom:to:into:startingAt:`
- `OrderedCollection >> addAll:`
- `OrderedCollection >> _addAllFromNsc:`
- `SequencableCollection >> copyFrom:count:into:startingAt:`

- Multi-threaded garbage collection
 - markForCollection, etc
- Multi-threaded Tranlog Replay
 - #restoreFromLogs:
 - Crash recovery from crash
- Multi-thread the Stone process
- Improved VM Performance

- Additional Special Classes
 - Candidates:
 - Date
 - DateTime
 - Time
 - Currency
 - ScaledDecimal
- Reduced Conflict (RC) Indexes on Collections



- Thread-safe C-interface (GCI layer)
- Support for Seaside
 - Need native support for continuations



Questions?

