GLASS: *A Share Everything Architecture for Seaside*

Dale Henrichs

8/28/2008

What is GLASS?

- GemStone, Linux, Apache, Seaside, Smalltalk
  - Persistence and scalability for Seaside applications
  - FREE for commercial use
    - http://seaside.gemstone.com/downloads.html

What is GLASS?

- GLASS
  - self-contained development environment for GemStone/S
    - Monticello for source code control
    - OminiBrowser-based development tools
  - Easy to move apps from Squeak to GLASS
  - FREE for commercial use

What is GLASS?

- Appliance
  - VMWare running an instance of GLASS
    - 3 Seaside VMs
    - 1 Maintenance VM
    - 1 (or more) development Vms connected to Squeak
  - FREE for commercial use

Share Nothing Architecture

- Scaling Story
  - Every time you need data, you hit the data base
  - Distributed computing and distributed data
    - add additional resources to meet demand
  - Well suited to stateless web servers and uncomplicated data models

Share Nothing Architecture

- Drawbacks
  - Scaling limited by available electricity
  - distributed data constrains model complexity

Share Nothing Architecture

- Smalltalk is not **Share Nothing**

Share Everything

- Smalltalk is
  - **Share Everything** inside the image
  - **Share Nothing** outside the image

GemStone/S

- GemStone/S is designed for
  - very large images
  - shared between multiple VMs
  - running on multiple computers
- GemStone/S is the perfect vehicle for doing **Share Everything** with **Share Nothing** scalability

Seaside Scalability Options

- Persistent Session State
- Optional _s and _k
- One Session per VM

Scalability – Persistent Session State

- Scaling Story
  - store/share Seaside session state in repository
  - deploy any number of VMs
  - round robin requests without session affinity

Scalability – Persistent Session State

- Drawbacks
  - request rate limited by commit rate
    - depending upon hardware limit kicks in at about 10-100 commits/second
  - scaling requires investment in more sophisticated hardware
    - **NOT** commodity hardware

Scalability – Optional _k _s

- Scaling Story
  - Avoid saving 'unnecessary' session state
    - reduce commit rate
  - Performance Potential
    - 7K requests/second
    - 128 VMs
    - 72 CPUs
    - 7 machines

Scalability – Optional _k _s

- Drawbacks
  - application must be changed (RESTful)

Scalability – One Session per VM

- Scaling Story
  - session state in temporary memory
  - VM dedicated to session for its lifetime
  - performance similar to *Optional _k _s*

Scalability – One Session per VM

- Drawbacks
  - need more characterization work
  - swap/real memory limitiations
    - need to characterize tradeoffs

GLASS

- A Seaside application written to run in Squeak can be ported to GLASS 'without modification'
  - Transparent Persistence
  - Transparent Scalability
    - persistent session state
    - one session per vm

Transparent Development

- Multi-image development support
  - auto commit
  - object log
  - debugging
    - breakpoints
  - profiling

- Demo

Futures

- GemStone 3.0
  - non-Tranlogged objects
  - Native code generation
  - Improved Exception Handling
  - Foreign Function Interface
- ...and Beyond?
  - sharding for GemStone/S

Questions?

- http://seaside.st
- http://seaside.gemstone.com
- http://gemstonesoup.wordpress.com