The BKNR Datastore

An introduction

Hans Hübner

Outline

- History, Motivation, and Goals
- Architecture
- Developer considerations
- Application Demo
- Experience, Status and Future

History and Motivation

- eBoy, a graphics artist group, wanted to have a new dynamic, graphics oriented web site
- In 2004, they agreed to have us program it for them in Common Lisp
- Persistence solution required
- Object database preferred
  - Better match from domain model to DB
  - Better agility for iterative development model
- Existing solutions too expensive
  - ACL/ObjectStore
  - cl-versant

Goals

- Transparent persistence
- Native Lisp
- High performance
- Lightweight deployment
- Open source

Why no database?

- Mismatch between OOP and RDBMS
- Schema maintenance with ORM painful
- Scalability depends on RDBMS knowledge and tools
- Using SQL is wasteful, both in terms of programmer and program execution time

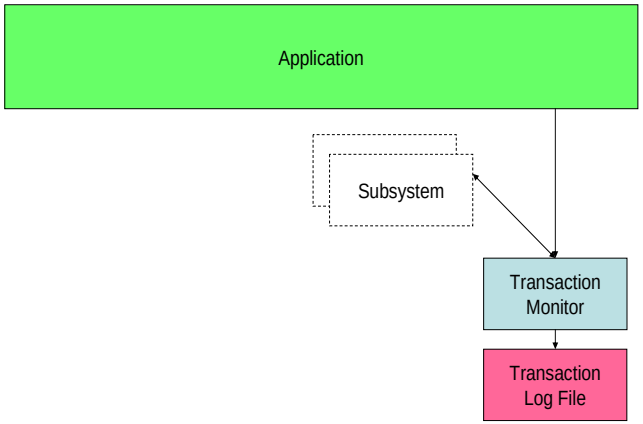
Predecessors & Influences

- cl-prevalence by Sven van Caekenberghe
  - Logging in XML
  - API influenced by Prevailer
  - No MOP integration
- cl-versant by KnowledgeTools
  - Used Versant as backend
  - Good MOP integration
  - Uses native Versant queries in Lisp syntax

Architecture

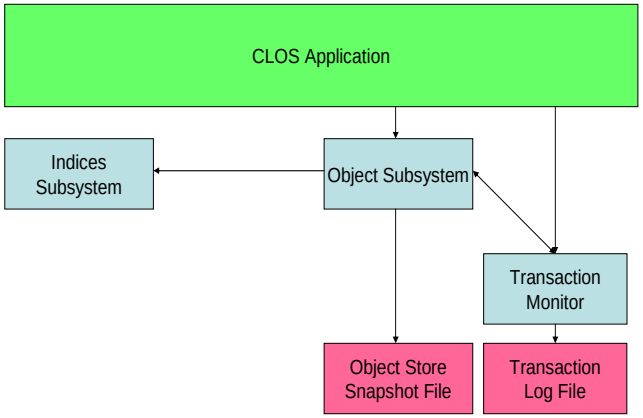
- All persistent data is kept in main memory
- All operations that change data are logged to a file
  - Name of transaction
  - Arguments
- Upon startup, the log file is replayed, restoring the persistent state
- Snapshots can be used to speed up recovery

Components



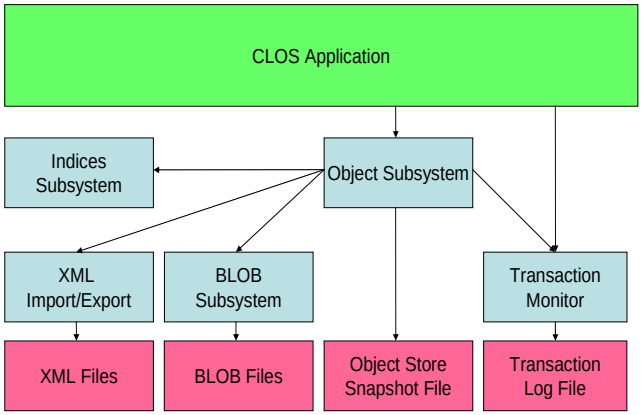
- Transaction logging mechanism
  - Supports named transactions and a variety of atomic Lisp data types
  - Log file in an extensible binary format
- Subsystem based snapshot and restore
  - The object store is one subsystem
  - Other subsystems can be added, should the need for non-object-based snapshots arise.

Components (contd)



- Object store subsystem
  - Client to the transaction mechanism
  - Persistent CLOS objects
- Indices
  - Used by the object store for class object tables
  - Can be used separately
  - Can be extended for special-purpose indices
  - Types: string-unique, skip-list, special-purpose

Compontents (contd)



- Blob subsystem
  - Supports file-based BLOBs
- XML based import and export
  - Uses the persistent object infrastructure
  - DTD based
  - Automatic export and import based on metadata added at the MOP level

Object subsystem explained

- Implementation is based on closer-mop
- Database schema defined by persistent class definitions
- All persistent classes share a base class and a meta class
- Slot write access outside of transactions is not possible
- WITH-TRANSACTION macro logs slot changes
- Each class has an object table holding its instances

Developer Considerations

- Instances that are no longer used must be explicitly dereferenced
- All mutating operations are synchronized by the transaction monitor. There is no write concurrency
- Transactions must be self contained, fully reproducible and not perform I/O
- Failures must be handled *within* the transaction

Application Demo

- Borneo Orang-Utan Survival foundation
- NGO supporting Orang-Utans in Indonesia
- Samboja Lestari project partially “sold” to sponsors
- Sponsors get their personal square meters assigned
- Early AJAX application
- Currently being extended into Google Earth

Queries

- Queries for persistent objects are written in Lisp
- No need to switch languages
- No need for fancy syntax layers to make query language bearable in Lisp

Performance

- Disk files only used for sequential read/write
- Not doing any random disk access (except for blog I/O)
- Memory overhead considerable (but is it?)
- Profiling considerably eased as no separate profiling tools for the database required

Compiler Portability

- Current deployments on cmucl-19c
- Trunk development and testing on SBCL and Clozure CL
- Trunk support for cmucl-19e planned
- Allegro CL has been used in the past, but not verified in a long time

OS Portability

- FreeBSD is our deployment platform
- Buildbot used to automatically build and test on FreeBSD, Mac OS X and Linux
- Development on FreeBSD and Mac OS X

Areas of Work

- Hot standby
  - As only transaction code may mutate data, running multiple servers with one master executing the changes may be used for hot standby
- Fork-on-Snapshot
  - Before snapshotting, use Unix to make a copy of the whole process to save time.
- Image Snapshotting
  - Optionally, allow for snapshotting to a Lisp image for faster startup.
- Rollforward and log tools
  - More tools for logfile analysis and rolling forward to specific time stamps might be helpful.

More areas of work

- Enhance packaging
  - Support development without BKNR's thirdparty tree
- Store code mutations
  - Instead of loading the code from the file system, load it from the store. Also put code changes to the store.
- Enhance XML subsystem
  - Interact with XML Schema or RelaxNG instead of DTDs
- Annotations for associations
  - Model association cardinality in object slots
- Write more and better documentation

Current status / Future

- Running production systems
  - quickhoney.com
    - 80,000 objects, 80 MB resident size with CMUCL
  - create-rainforest.org
    - 350,000 objects, 450 MB resident size with CCL
- Major relaunches for both in the works
- No next new project yet
- Best-effort support for third parties

Libraries used / Acknowledgements

- Contributions
  - David Lichteblau
  - Kilian Sprotte
  - Manuel Odendahl
- Libraries
  - cl-ppcre, cl-interpol
  - cxml

Availability

- bknr and most application code is available under a BSD-style open source license
- Repository instructions are available at http://bknr.net/
- If you use the software, we'd like to hear from you!

Thank you for your attention!

Questions?