The BKNR Datastore

An introduction

Hans Hübner

Outline

- History, Motivation, and Goals
- Architecture
- Applications
- Experience, Status and Future

History and Motivation

- In 2004, eBoy, a graphics artist group, wanted to have a new dynamic, graphics oriented web site
- Dynamic – That's a task for (Common) Lisp
  - Took quite some convincing
  - „You can learn Lisp, too!“ *cough*
- Object database wanted
  - Better match from domain model to DB
  - Better agility for iterative development model
- No affordable product on the market
  - cl-versant had licensing issues
  - Franz´ offerings likewise

Goals

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

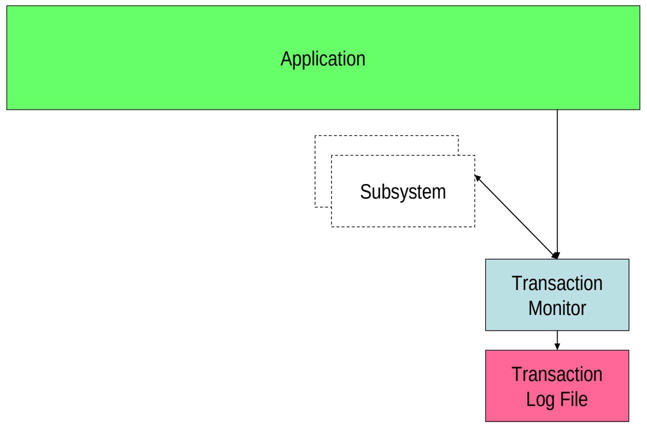
Predecessors & Influences

- cl-versant by KnowledgeTools
  - Used Versant as backend
  - Good MOP integration
  - Uses native Versant queries in Lisp syntax
- cl-prevalence by Sven van Caekenberghe
  - Logging in XML
  - Clumsy API
  - Too much work needed, but we stole the idea

Principle of Operation

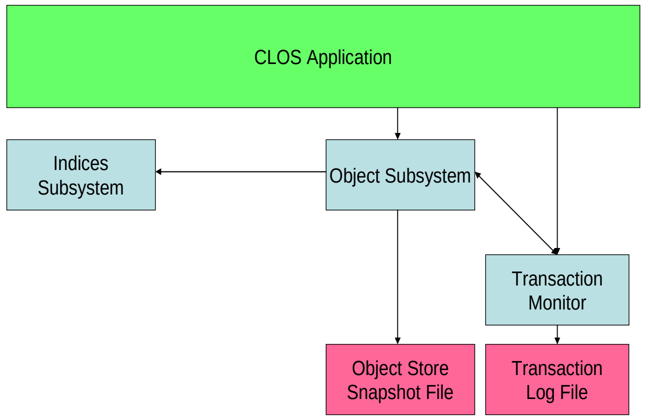
- All persistent data is kept in main memory
- All operations that change persistent data are logged to a file
  - Operation name and argument values
  - Change size ‹› log volume
- To recover the persistent state, the log file is replayed
  - Recovery takes as long as original execution
- Snapshots for faster recovery

Architecture



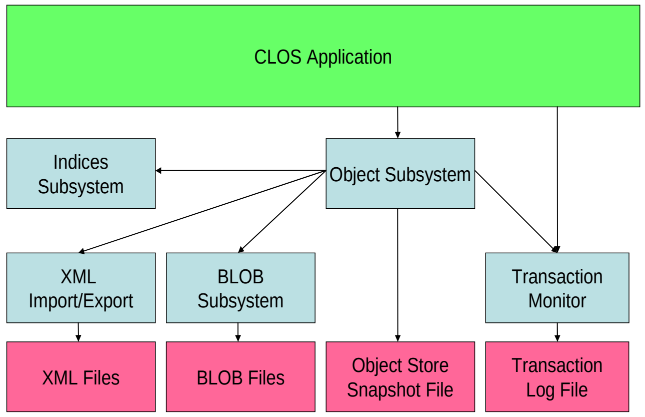
- Transaction logging mechanism
  - Supports named transactions and a variety of atomic Lisp data types
  - Log file in an extensible binary format
- Subsystem based snapshot and restore
  - The object store is one subsystem
  - Other subsystems can be added, should the need for non-object-based snapshots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

Architecture (contd)



- Indices
  - Used by the object store for class object tables
  - Can be used separately
  - Extensible
- Object store subsystem
  - Client to the transaction mechanism
  - Persistent CLOS objects

Architecture (contd)



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

Object Subsystem

- 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
- Named transactions must handle failures
- Anonymous transactions roll back on error
- Read consistency does not come for free

Why no RDBMS?

- 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
- Adds deployment requirements

But I want 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
- But: FIND and EVERY are SOMEtimes clumsy

Performance

- Disk files only used for sequential write
- Not doing any random disk access (except for BLOB I/O)
- Memory overhead considerable (but is it?)
- Easy profiling eased as Lisp tools can be used
- Current implementation does not scale to multiple processors

Applications

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

Applications

QuickHoney™

- Illustration company
- Easy updates
- Ajax / JSON
- Server side image manipulation through cl-gd
- RSS/Twitter

Current status / Future

- Running production systems
  - quickhoney.com
    - 150,000 objects, 209 MB resident size with CCL
  - create-rainforest.org
    - 728,000 objects, 317 MB resident size with CMUCL
  - ruinwesen.com
    - Just recently launched, SBCL/Linux
- Major relaunches for create-rainforest.org in the works
- Best-effort support for third parties

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
- Annotations for associations
  - Model association cardinality and ownership in object slots
- Concurrency support
  - Requires per-thread copies of indices
  - Multiple stores?
  - STM?
- More safety checks
  - Disallow calling of certain functions in transactions
- Write more and better documentation

Availability

- bknr and most application code is available under a BSD-style open source license
- Repository instructions are available at http://bknr.net/

Compiler Portability

- Current deployments on cmucl-19c and SBCL-1.0.20
- 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

Libraries used / Acknowledgements

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

Concluding Remarks

- Lisp is great!
- Refactoring and substantially changing the architecture has not been a problem
- The MOP makes implementing advanced features easy, but metaclasses do not compose well
- Concurrency is an unsolved problem
- No general library problem, but getting them is a problem
- I'd write it again, without define-persistent-class though

Thank you for your attention!

Questions?