manardb: fast memory mapped objects in Common Lisp

John Fremlin

http://john.freml.in

2012 October 6

Contents

Manardb is a memory mapped object database portable across many common Lisps on Linux.

Outline

In memory objects

- (defmmclass my-object () ...)
- retrieve-all-instances
- persistent
- transactions
- garbage collector, etc.

Why?

- High data rate, requiring really fast updates: millions of objects all updated every few minutes.
- AllegroCache was too slow.
- No way to use MySQL unless we batched up huge database rewrites in single requests.
- Lisp GC was killing us.

Special features

- Memory mapped.
- Low startup time.
- External to the GC in the host Lisp.
- But almost like dealing with normal Lisp objects.
- Two garbage collection strategies.

Special features

- Memory mapped.
- Low startup time.
- External to the GC in the host Lisp.
- But almost like dealing with normal Lisp objects.
- Two garbage collection strategies.

Special features

- Memory mapped.
- Low startup time.
- External to the GC in the host Lisp.
- But almost like dealing with normal Lisp objects.
- Two garbage collection strategies.

Example

```
(use-package 'manardb)

(use-mmap-dir "/tmp/")

(defmmclass person ()
  ((name :type 'string :initarg :name)))

(make-instance 'person :name "John")

(retrieve-all-instances 'person)
```

Outline

Memory mapping

- One file per object type.
- Some special object types: arrays, lists, strings.
- Simultaneous access from multiple threads and multiple processes.
- Transactions: copy everything to a temporary directory, perform operations. If successful, move over to the main store (atomically).
- Lock files with versions.
- Could be really efficient with reflink.

Memory mapping

- One file per object type.
- Some special object types: arrays, lists, strings.
- Simultaneous access from multiple threads and multiple processes.
- Transactions: copy everything to a temporary directory, perform operations. If successful, move over to the main store (atomically).
- Lock files with versions.
- Could be really efficient with reflink.

Meta-object protocol

- Defines a new meta-class: mm-metaclass.
- Objects are represented to the host system as an unsigned integer.
- Access actually uses pointers to mmap'd regions.
- All normal access methods still work: slot-value, etc.
- Inheritance and function overloading work as if these were normal objects in the host system.
- Can mix normal slots with memory mapped ones!

Meta-object protocol

- Defines a new meta-class: mm-metaclass.
- Objects are represented to the host system as an unsigned integer.
- Access actually uses pointers to mmap'd regions.
- All normal access methods still work: slot-value, etc.
- Inheritance and function overloading work as if these were normal objects in the host system.
- Can mix normal slots with memory mapped ones!

Bugs in production Lisps

- Tested on Allegro, Lispworks, SBCL, and ClozureCL.
- Found bugs in all of them - typehints, broken caching, etc.
- MOP support is tricky!

Garbage collection

- Traditional mark and sweep.
- Walk all objects from the class definitions, walk arrays, etc. Store all edges in the object reference graph in memory.
- Now compact and copy all reachable objects.
- Rewrite all references between objects.
- Terribly inefficient!

Garbage collection

- Traditional mark and sweep.
- Walk all objects from the class definitions, walk arrays, etc. Store all edges in the object reference graph in memory.
- Now compact and copy all reachable objects.
- Rewrite all references between objects.
- Terribly inefficient!

Garbage collection

- Traditional mark and sweep.
- Walk all objects from the class definitions, walk arrays, etc. Store all edges in the object reference graph in memory.
- Now compact and copy all reachable objects.
- Rewrite all references between objects.
- Terribly inefficient!

Shortcut garbage collection

- No need to store full reference graph.
- Assume that there are no repeated references to the same object except for some restricted set of object types.
- Just sweep from the roots and copy all reachable objects whenever they're seen.
- Much faster, much less memory!
- rewrite-gc

Outline

Conclusion

It's possible to make a low-level extension to the Common Lisp object system that's portable across implementations!Use manardb.

Conclusion

It's possible to make a low-level extension to the Common Lisp object system that's portable across implementations!Use manardb.