AutoFetch
Ali Ibrahim and William Cook
University of Texas at Austin
Presented at ECOOP 2006

Object Persistence
- Transparent access to persistent objects
  - Load as needed: “Object faulting”
- Examples
  - Object-relational mapping tools
    - EJB, JDO, TopLink, Hibernate, OJB, and many more
  - Object databases
    - Versant, db4o, Gemstone, etc.
  - Orthogonally Persistent Programming Language
    - PJama, OPJ, etc.
- Object faults are expensive
  - Connection overhead > query execution

Example Data Model

Queries + Traversal
String q = "from Employee e
  where e.overtime > 100"
for (Employee emp: runQuery(q)) {
  Employee mgr = emp.getSupervisor();
  sendEncouragingEmail(emp.getEmail(),
    emp.getName(),
    mgr.getName());
}

Should prefetch managers

Manual Prefetch
String q = "from Employee e
left outer join fetch e.supervisor
where e.overtime > 100"
for (Employee emp: runQuery(q)) {
  Employee mgr = emp.getSupervisor();
  sendEncouragingEmail(emp.getEmail(),
    emp.getName(),
    mgr.getName());
}

Subtle dependency...
... must be maintained if
code change

AutoFetch Goals
- No programmer provided
  prefetch hints or
directives
  - Programs easier to write
  - Programs more modular
- Performance as good as or better than
  programmer
  provided prefetch directives

Traversal profiling
Each query produces a program traversal.
Program traversals are aggregated into
traversal profiles.

AutoFetch
- AutoFetch transparently
  prefetches objects
  based on program history
- AutoFetch can be broken into 3 aspects:
  - Traversal Profiling
  - Query Classification
  - Traversal Prediction
- Implemented as extension to Hibernate 3.0
  - Object proxies used to monitor
    association traversals
- Idea not specific to Hibernate or ORDBMS
Query Classification

- Identify similar query invocations
  - Traversal Profiles aggregate similar invocations
  - Apply prefetch corresponding to previous traversals
- Use the program state to classify queries
- How about?
  - query string
  - line number where query invoked

Classifying by Stack Trace

- Query invocations with same stack trace are classified together
- Benefits of stack trace:
  - easy to compute
  - finite number of stack traces
- Stack trace predicts future control flow

Traversal Prediction

- Query class $\rightarrow$ traversal profile
- Add association paths with probability greater than a threshold as prefetch specifications

Torpedo Benchmark

- Measures:
  - number of queries
  - Web auction application
  - 17 use cases
- AutoFetch is a fast as hand optimized
- ...with simpler code

OO7 Benchmark

- Measures:
  - traversals, queries, and updates
  - Based on CAD applications
  - AutoFetch executes approximately 100 times fewer queries

Related Work

- Prefetch for Object Databases
  - Object clustering
  - Object prefetch based on pattern analysis of object requests (Curewitz 93, Knafla 98, Palmer 91)
- PrefetchGuide (ORM)
  - Bernstein VLDB 1999, Han Infor. Sciences 2003
  - Optimizes traversals within a single query
  - Looks for recursive or iterative patterns

How is AutoFetch different?

- Disadvantages:
  - Does not optimize initial query executions
- Use AutoFetch + PrefetchGuide
- Advantages:
  - Best performance:
    - AutoFetch: 1 query
    - PrefetchGuide: at least 2 queries
  - Can prefetch arbitrary object graphs
  - More data for prediction
Future Work

- Improving benchmarks for OPA.
- Using AutoFetch ideas for general prefetch in distributed applications.
- Optimizing other aspects of OPA such as memory management.

Conclusion

- Predicts correct prefetch directives based on past program query executions
- Uses dynamic profiling
- Encourages more modular programs
- General technology for object persistence architectures