Object Oriented Database Systems

A Problem

Impedance Mismatch

- RDBMSs have their own
  - naming conventions
  - type system
  - conventions

Impedance Mismatch

- PLs have own notion of namespace, types, and return conventions
- Translating back and forth was a hassle, “Impedance mismatch problem”

Example

```
Struct Part {
  Int num;
  Char* name;
  Char* color
} part;
```

```
Define cursor P as
  Select * from Part
  where pno = 16;
```

```
Open P into part Until no-more{
  Fetch P(part.num = pno,
          part.name = pname
          part.color = pcolor)
  }
```

Another Problem

Trees in ML

```
type 'a tree =
   Leaf of 'a
 | Node of 'a * 'a tree * 'a tree
```

Trees in Java

```
class Tree {
  Object elem;
  Tree left;
  Tree right;
}
```

(Int) Trees in a RDBMS

Element(INT key,INT val,INT left, INT right)

‘a trees in a RDBMS

Element(INT key,INT val,INT left, INT right)

Values(INT key, BLOB value)

Height in ML

```
fun height (Leaf _) = 0
  | height (Node _,l,r) =
      1 + max(height l,height r)
```

Height in Java

```
int height(){
  return
   1 + max(left.height(),right.height);
}
```

Height in SQL

Impossible

Brouwer Ordinals in ML

```
type ord =
    Zero
  | Succ of ord
  | Limit of nat -> ord
```

Ordinals in Java

```
class Ord {
  some horrible anonymous class kludge
}
```

Ordinals in a RDBMS

Hmmmm...

code pointers stored in tables?

Lesson

- Data can be complex!
  - Recursive (Lists)
  - Polymorphic (Lists)
  - Active (Closures)
  - Large (Audio, Video)

A (Partial) Solution

OODBMS/ORDBMS

- Attempt to add support for
  - User defined data types
  - Inheritance
  - Object Identity

User Defined Types

- New datatypes (eg. trees)
- Functions that operate on new types
  - predicates are handled on the server instead of the client.
  - CREATE FUNCTION ...
  - DB/2 allows SQL,C++, Java

Inheritance

- CREATE TYPE t2 UNDER t1
- t2 “inherits” the attributes and methods of t1
- The usual dynamic binding

Inheritance, cont.

- Table inheritance as well
  - CREATE TABLE table1 OF TYPE t1
  - CREATE TABLE table2 OF TYPE t2 UNDER table1
  - Arbitrary trees of tables, called the "collection hierarchy"
  - Queries run over all descendants

Object Identifiers

- OODBMS store unique identifiers for objects
- Used for “pointer dereferencing” and optimized equality checking

(Notice the ligature “fi” in the title...)

Trees in OODBMS

- CREATE DATA TYPE TREE (val = INT, left = REF TREE, right = REF TREE);
- CREATE FUNCTION height(TREE) RETURNS INT AS EXTERNAL NAME 'tree.class' LANGUAGE 'java';
- Concern: what if height modifies its arg?

Ordinals

- You can even hack ordinals in, but it's better to use a package like JDO

```
<jdo>
    <package name="">
        <class name="Ordinal" />
     </package>
</jdo>
```

Implementation Challenges

Large Objects

- Large structured objects complicate data layout
- In practice, many objects are larger than a page
- "Bulk" attributes (like lists) can grow arbitrarily, requiring flexible disk layout

Indexing New Types

- RDBMS index structures support only equality conditions (B+ trees)
- Each new datatype may have special structure that allows an efficient index

Indexes Cont.

- Solution1: Allow user to create access methods with external code
- But: External index is not protected for concurrency and recovery.

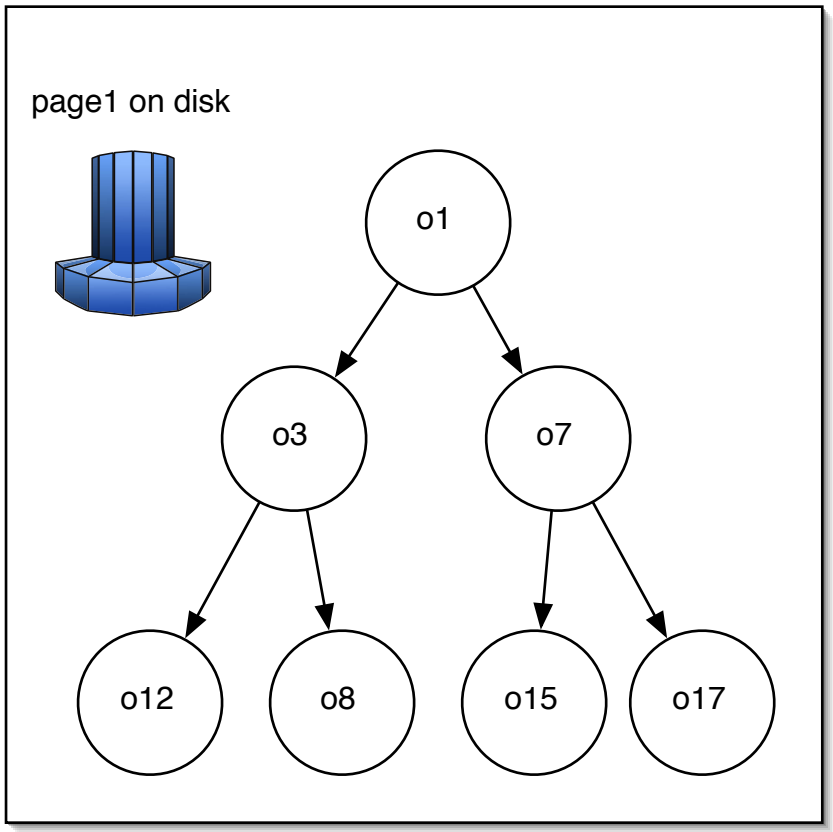
Indexes, cont.

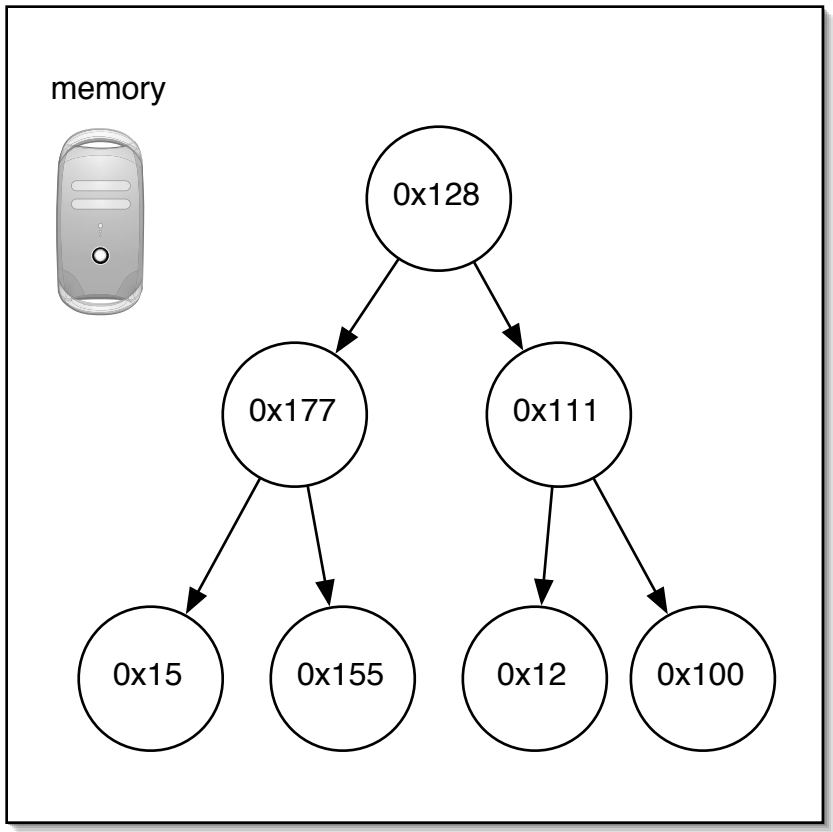
- Solution2: OODMBS provides a ‘template’ index structure that is more or less general.
- Most tree index structures can be implemented easily.

Query Processing I

- Problems with our model
  - Security of user defined functions
  - Pointer Swizzling
  - Query optimization

Pointer Swizzling





Query Optimization

- Painful with new indexes
- Must register with the optimizer
- Difficult to estimate the cost of external methods.

ObjectStore

Objectives (ha ha)

- Smooth integration with PL (C++)
- Unified interface to persistent and transient data
- Object access speed “equal” to an in-memory pointer dereference

Key Theoretical Idea

- Persistence is not part of the type system

Example

```
database *db;
persistent<db> dept* cs;
db = database::open("cs");
transaction::begin();
gradstudent *sean = new (db) gradstudent("sean");
cs -> add_student(sean);
sean->salary = 100000000;
transaction::commit();
```

Key Imp Idea I

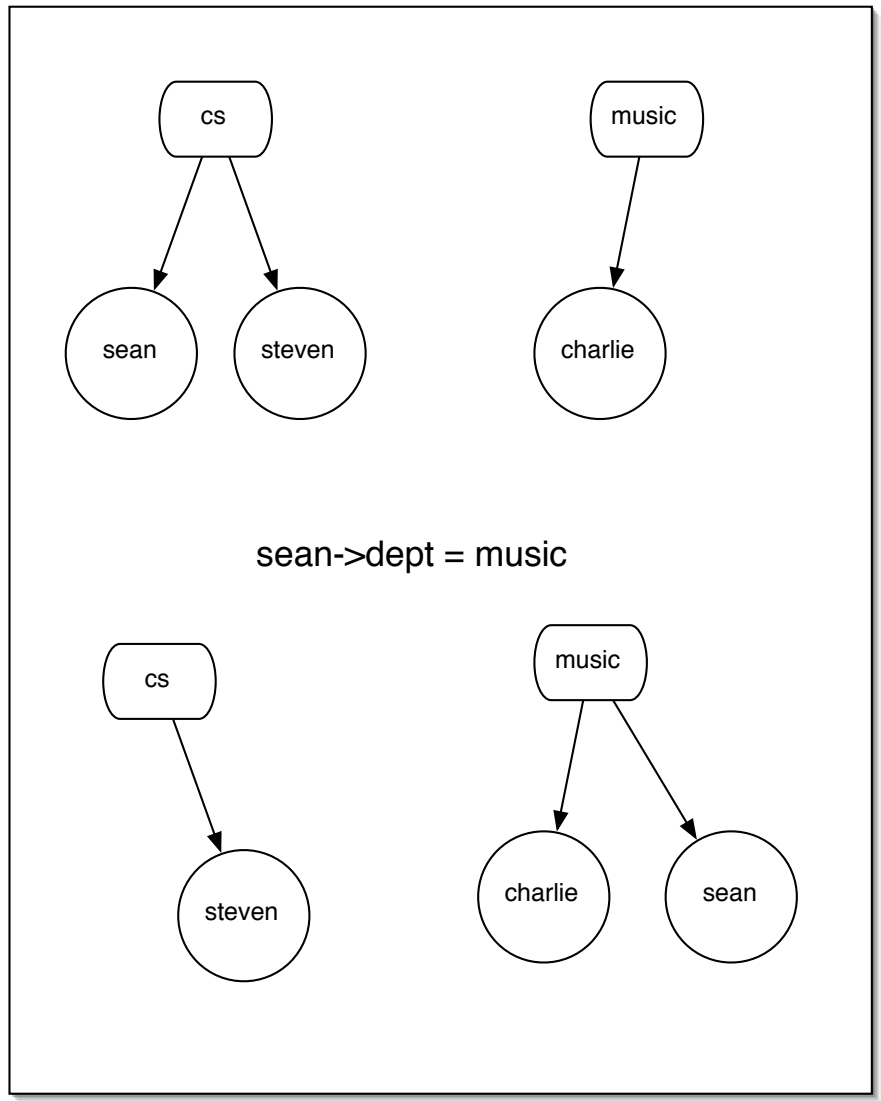
Bidirectional Relationships

(Actually stolen from the Entity-Relationship data model)

Relationships

- Maintain database integrity
  - Deletions delete entire subtree, analogous to garbage collection
  - Updates manage pointers implicitly

Example


cs
music
sean
steven
charlie
sean->dept = music
cs
music
steven
charlie
sean

Minor Imp Ideas

- Embedded query language
  - Actually, this is much nicer than the usual SQL translation
  - Avoids the semantic mismatch of SQL and the target language.

Query Language Example

```
d->employees[:salary >= 100000 :]

all_employees
  [: dept->employees
     [:name == 'Fred':]
  :]
```

Minor Imp Ideas

- Versioning system (CVS for objects)
  - Useful for enormous engineering objects that get passed from dept to dept.
  - Alternative to concurrency control

Key Imp Idea II

Memory Mapping

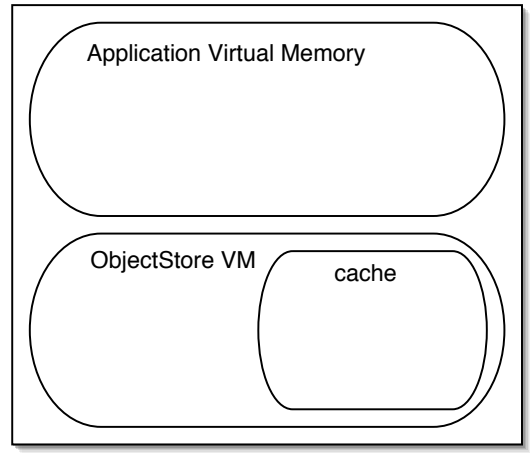
Implementation

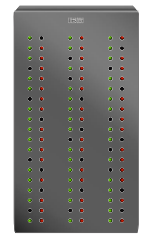
- Recall goal 3: Persistent pointer dereference is "as fast" as ordinary pointer
  - Fundamental OO operation
  - Ordinary pointers must be able to serve as references to both transient and persistent data

Imp II

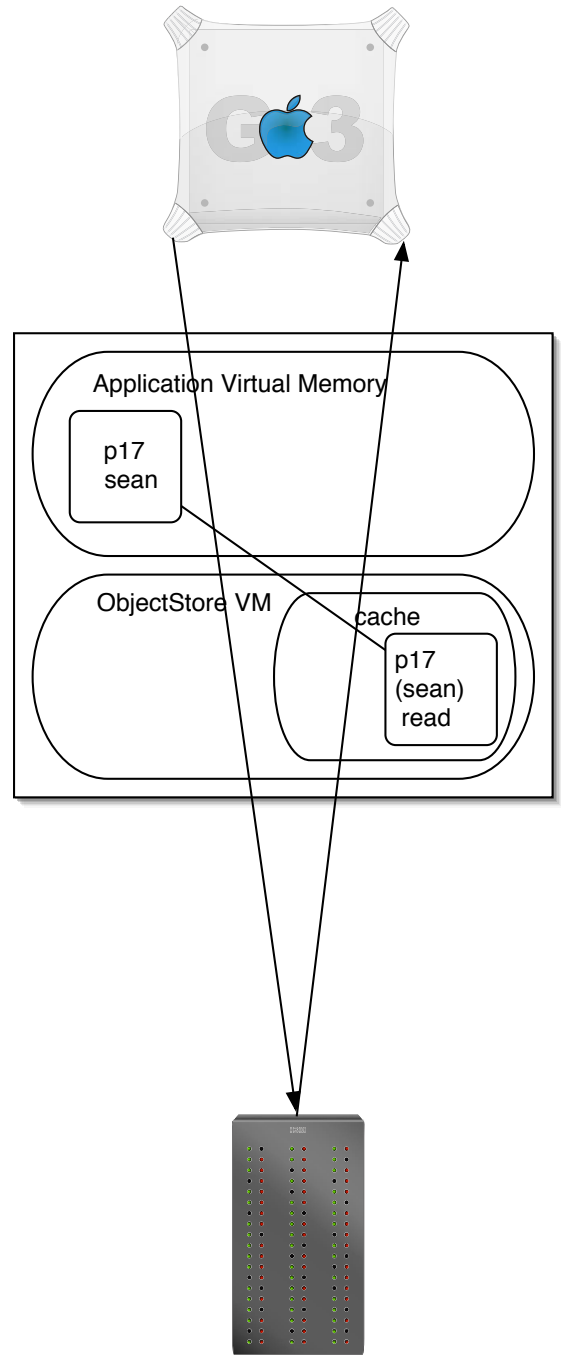
- Want to avoid runtime checks for object presence
- What if the object isn't in memory?
  - Solution: use the OS virtual memory
  - Can set the page protection (no access/ read only/ read-write)

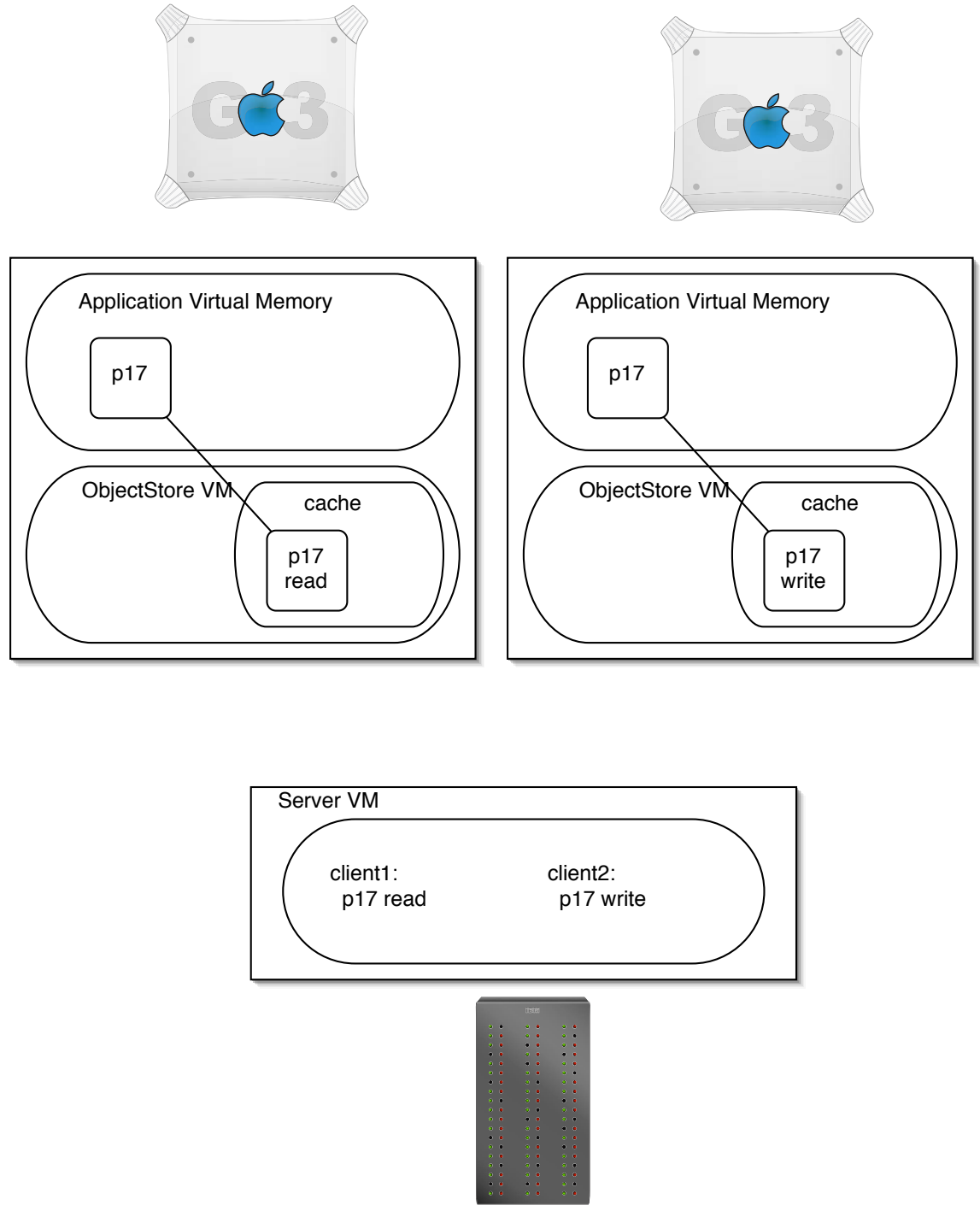
int salary = sean->salary;

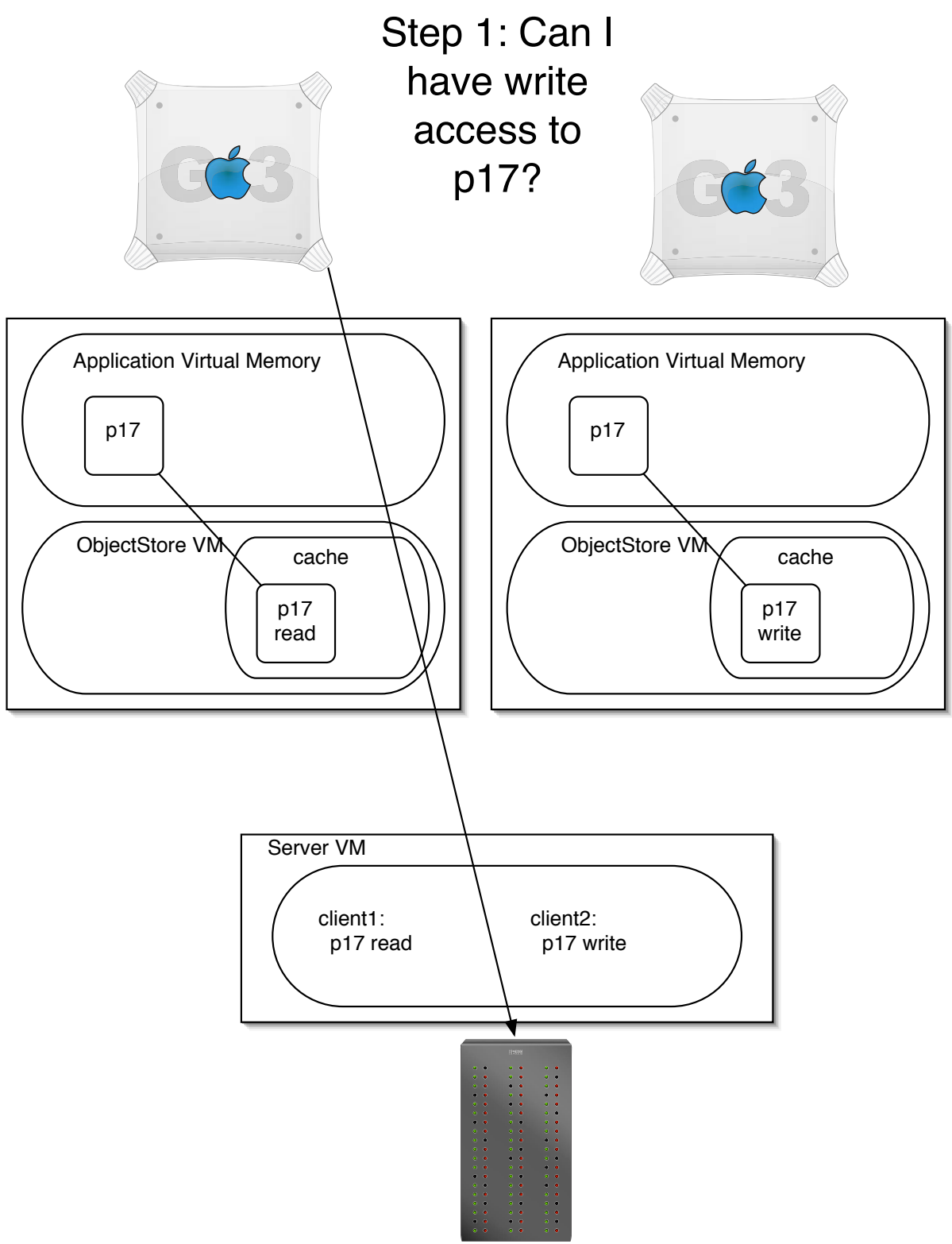


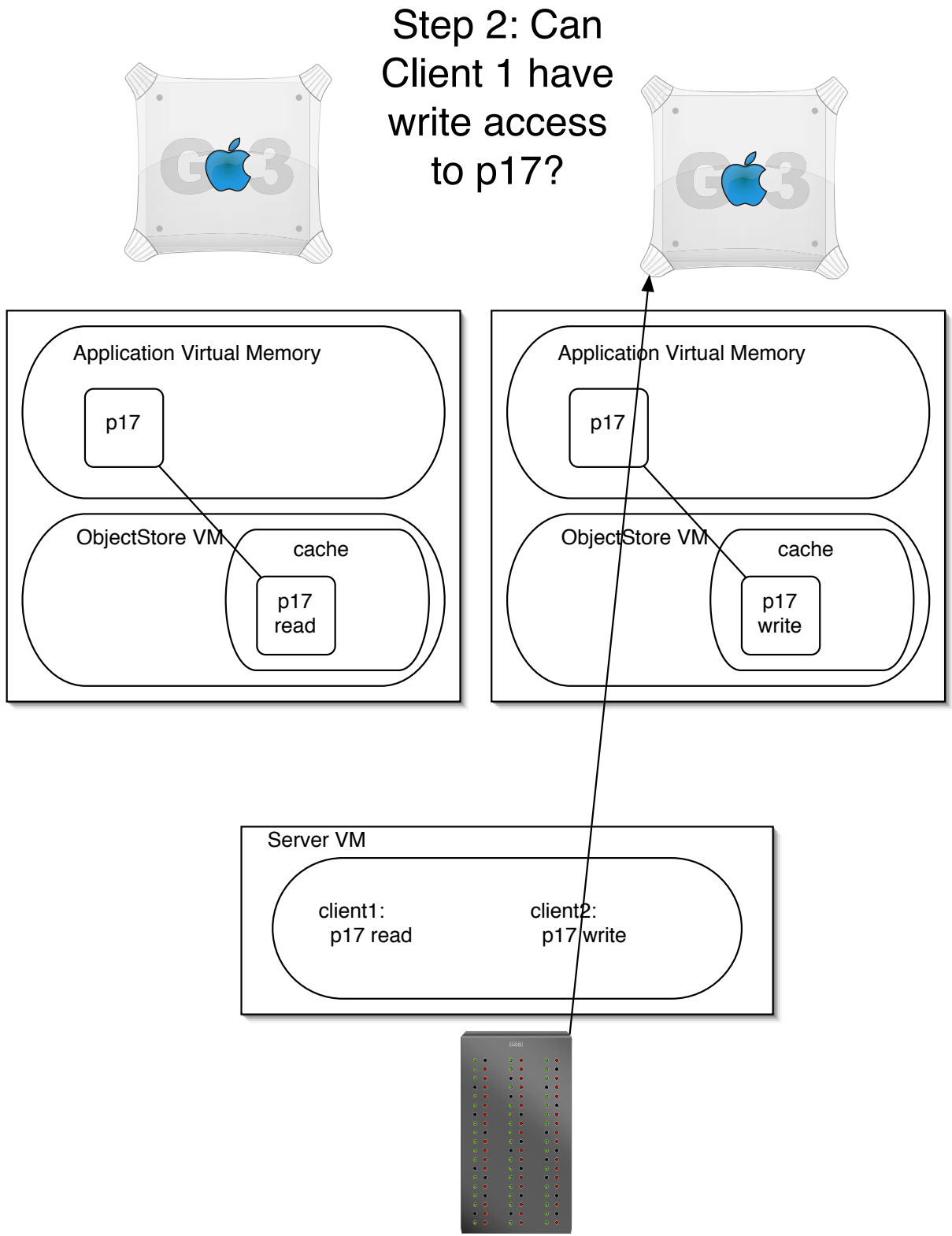


int salary = sean->salary;

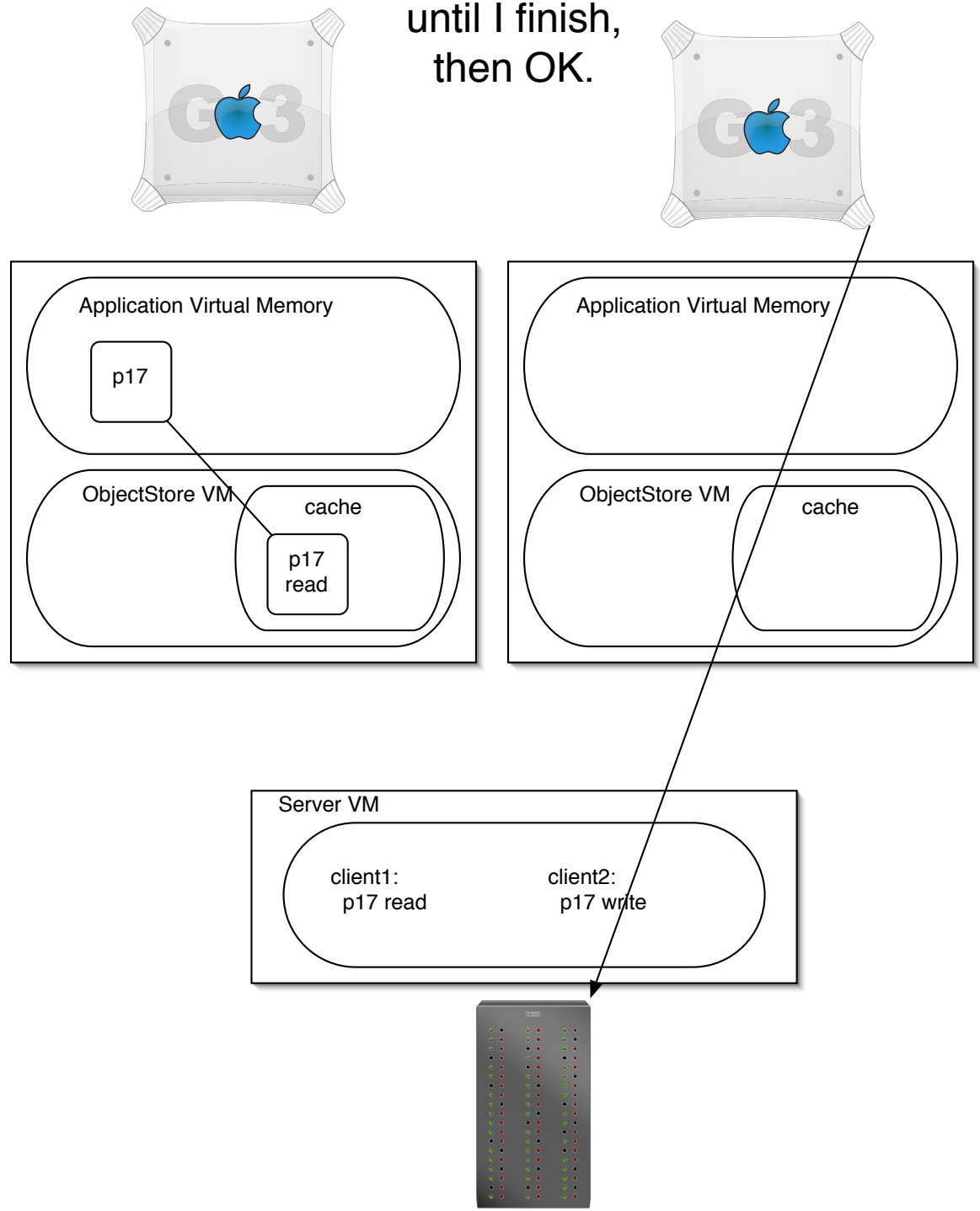


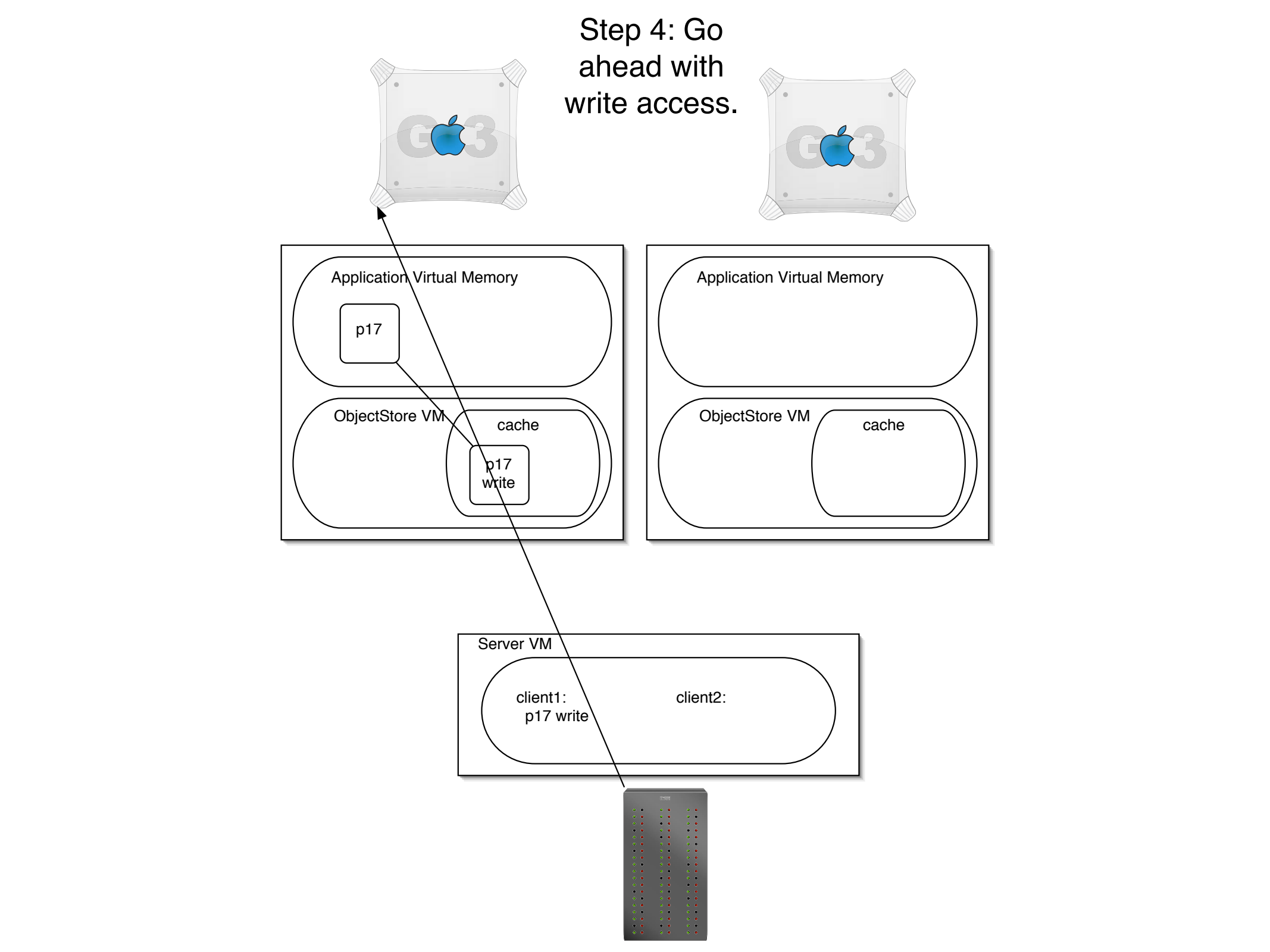

Application Virtual Memory
p17
ObjectStore VM
cache
p17
read
Application Virtual Memory
p17
ObjectStore VM
cache
p17
write
Server VM
client1:
p17 read
client2:
p17 write


Step 1: Can I have write access to p17?
Application Virtual Memory
p17
ObjectStore VM
cache
p17
read
Application Virtual Memory
p17
ObjectStore VM
cache
p17
write
Server VM
client1:
p17 read
client2:
p17 write


Step 2: Can Client 1 have write access to p17?
Application Virtual Memory
p17
ObjectStore VM
cache
p17 read
Application Virtual Memory
p17
ObjectStore VM
cache
p17 write
Server VM
client1: p17 read
client2: p17 write

Step 3: Wait until I finish, then OK.




Step 4: Go ahead with write access.
Application Virtual Memory
p17
ObjectStore VM
cache
p17 write
Application Virtual Memory
ObjectStore VM
cache
Server VM
client1: p17 write
client2:

Accessing Objects

- CPU executes regular load instruction
- hardware detects access violation, signals memory fault
- ObjectStore retrieves the page, places it **in the client's cache**
- Tells OS to set protection read-only
- Restarts memory reference

Accessing II

- writes signals access violation
- ObjectStore tries to upgrade lock to read-write

Transaction Management

- Pages can reside in the client cache without being locked
- Cache coherence problem
- Cache pages are marked as shared or exclusive

Trans. Management II

- When a page is requested, server checks to see if modes conflict.
- If they do, it requests holding client to release.
- If page is locked, client tells server to wait
- After commit or abort, client removes page from cache

Imp Idea III

- Divide disk into segments
  - User can have direct control over disk layout
  - Seems like a hassle...

Why did OODBMS Research Die?