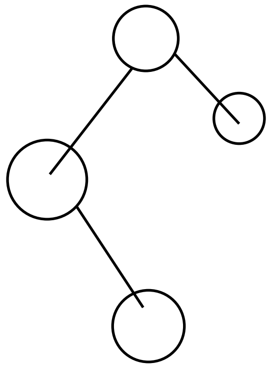
Mneme- Persistent Object Store

- **Object Structure**



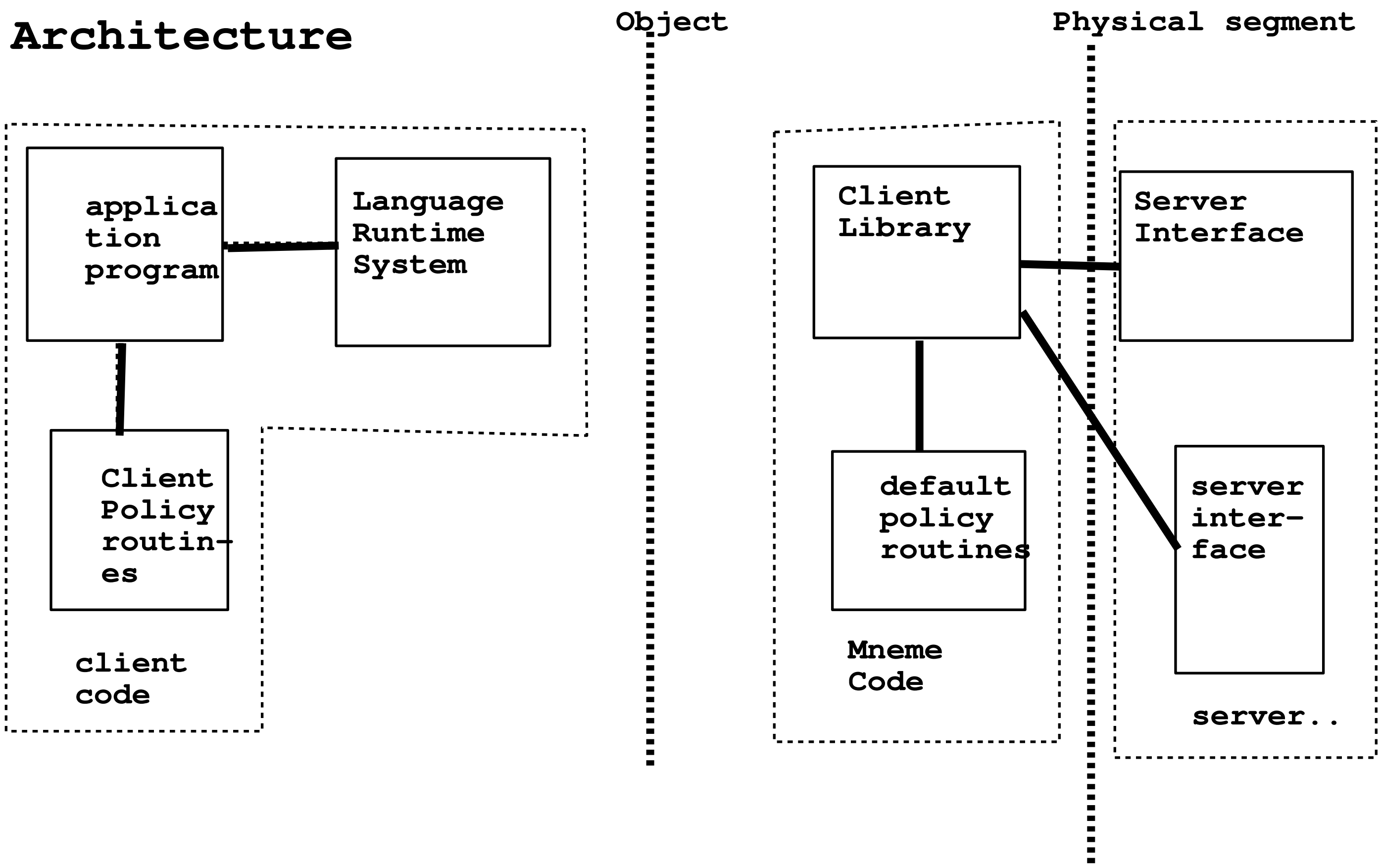
Objects are the nodes of a directed graph. egdes are pointers.
no typing and invocation mechanism in object storage

- **Perfamance**

if we use a separate strage manager like DBMS, then it will not be time efficient.

Other issues
portibility, extensibility, distribution, subcollections etc.

Architecture



Much of the detailed object semantics is to be implemented within the user's process and address space though storage facilities are provided by OS or a different server on a differnt machine.

issues: Safety, Concurrency, Buffering

Safety: Large granules less perfomance penalty
Concurrency: more than one level of granulity
Buffering: private to each client

Concepts

Objects

Object identifiers
Object_ids are enumerable-> helps in gc (garbage collection)
each object has a few attribute bits & a size
no type notion-> some bytes can be used as type/class
neither rich object semantics nor rich object relationship

Handle

A handle provides efficient manipulation of internals of a resident object.
Handle contains ptr to object, object_id, datasize and many other information. It saves time because object header can be interpreted into some bits and can be kept into the handle, direct memory access to the resident object is also possible. Creation and deletion od an handle delimit the time during which a data-object is accessed and must be available to the application.

Files and locality of ids

Every Mneme object resides in a file. Each file contains one or more physical segment. Files should be resonable unit of backup recovery, garbage collection etc. Objects in the same file point each other by Object_ids. Object can also point to an object in a differnt file. Object points to a forwarder object in the same file which contains a field indicating forwarding protocol number. Forwarding protocol number is mapped to a forwarding routine and this routine takes cares of forwarding to another file. Object_ids in the file are differnt from names of objects external to the file. This permits gc, object-reassignment, object deletion etc.

Pools and Strategies

Each Mneme object is associated with exactly one pool and that pool determines the policy under which the object should be managed. A management strategy is a vector of pointers to the routines of individual policy decision. Strategies can be dependent on specific pool variables called attributes. If an object fault occurs the object should be taken from server, but object fault policy routine determines prefetch-policy, buffer management, locking etc.

Transaction

Mneme uses serializability to maintain co-operation, but it provides non-serializable primitives for those situations where serializability is too limiting. A Mene store session is a period of interaction with store. A transaction is a unit of work, concurrency control, consistency within a seession.
--Rules
A set of committed transaction is gauranteed to be serializable and each transaction is gauranteed to be all-or-nothing.
--Lock
Lock on objects and different granuality is important. Locks are not generally expilicit, they are implicit in pool stragey routines.
--Mutual exclusion
When a client acquires an object-handle Mneme gives the client exclusive acquisition to the client until the release of the handle. This strategy is called volatile pool
--Notification
Notification events (abort, commit) end up with calling specific subroutines in strategy
----crush resilence
Nmene uses POSTGRES storage manager techniques for crush recovery. It mainly uses global logs segnment of data for each physical segment. Localized log information can be used for volatile pool

INTIAL PROTOTYPE

Limitations

1. 2^20 objects /file and 1024 physical segments/file
2. No multiuser-support, distribution or resilence(recovery)
3. Multiple files are allowed but cross-refenrences are not
4. Prototype supplies a small number of builtin object-formats rather than r elying on the client to deal with object formats and id-enumeration
5. no beffer management

Client-Interface

--client gets a uniform result code and a uniform error code when it returns from server routine calls
--All dynamic, visible structures are allocated and freed by client.
MnObjectCreate(poolid, nslots, nbytes, attrs, near, hnd, id)

MnObjectCreateptr(poolid, nslots, nbytes, attrs, near, ptr, id)

MnObjectDestroy
MnObjectrelease
MnIdMod

In direct pointer interface client can visualize the disk format.

Pool and Strategy Opreations

MnPoolCreate (fileid, Stratid, attr)
Only one startegy now.

FileOperation

mode-> sessionwise colletion of options. options are attributes attached to pools. Each operation cab examine options related to it. Each file has a root

Miscellaneous
statistics and accounts

IMPLEMENTATION

Language C, OS Sun OS or Ultrix or Vax/VMS

-Client and Persistent Indentifiers

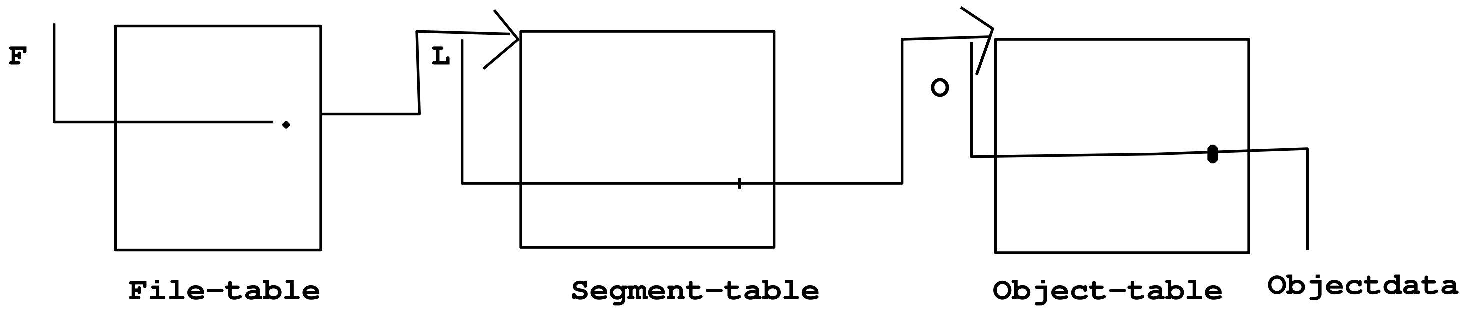
Same object id can be in more than one file. We have to disambiguate it.

Tag	File-bits	Number-of-files	Object-id-bits	Objects/file
00	11	2048	14	16384
01	8	256	17	131072
10	5	32	20	1048576
11	2	4	23	8388608

2^14 id, 2^13 files

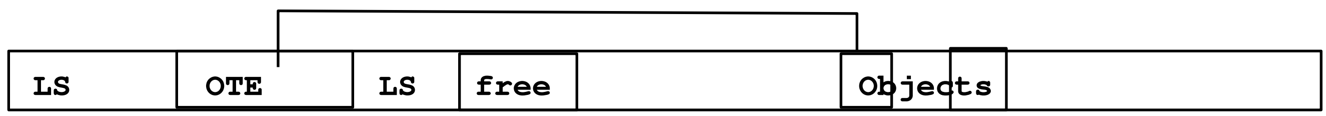
-Logical segment

A set of objectids that have same highorder bits be located in the same physical segment. So not the mapping from objects to physical segments rather mapping logical segments to physical segments. mapping table is smaller. Main memory can accomodate it.



-Physical Segment

When an object fault occurs we always retrieve a complete physical segment. physical segments has LS, OTE, Objects in it. Each physical segment belongs to exactly one pool. Physical segment implements logical segments along with providing clustering, trannsfer, update etc.



-Server interface

server interface includes a notion of file. Client sees it as file.

PERFORMANCE

Comparison with Exodus Storage Manager (ESM). The work for testing performance was constructing traversing and saving of a tree.

Creation of a node

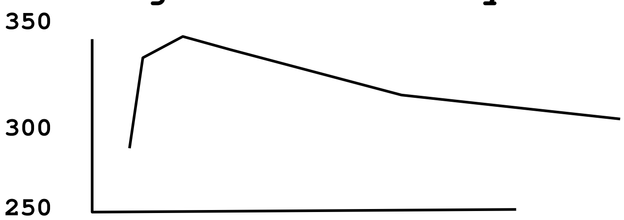
	C, custom	C, malloc	Mneme, Handle,	Mneme, direct
μsec	9.0	11.4	37.8	33.1

EMS was 4 times slower

Saving objects

Upto 8 MB storage was done
diskrate 360 to 370 KB/sec
Mneme   321 KB/sec
ESM   343 KB/sec

ESM objects were 12 bytes and Mneme objects were 4 bytes



Fetching Objects(upto 2 MB)

Mneme 279 KB/sec
ESM   220-230 KB/sec

Locating and manipulating Resident Objects

Reading and updating of slots, bytes, integers
time for perobject traversals

Nonpersistent C	11.7
Mneme , ptr	20.2
Nmene , handle	22.2
ESM	66

	Reading			Updating		
Software	slot	byte	integer	slot	byte	integer
NonpersistentC	.41	.20	.32	.44	.51	.52
Mneme	.42	.20	.33	.44	.51	.50
Mneme, handle	.83	.25	.51	1,06	.62	.72