Outline

- Review of distributed systems (cont)
- Previous lecture:
  - Stateless (NFS) vs Statefull (AFS)
  - Block level (traditional NFS) vs File level (traditional AFS) caching
  - Delayed write vs write through policy

File operations and consistency semantics for AFS

- Each client provides a local disk cache
- Clients cache entire files (AFS3 allows blocks)
  - Large files pose problem w/ local cache and initial latency
- Clients register **call back** with server & server notifies clients on a conflict read-write conflict to invalidate cache
- On close, data is written back to the server (last writer wins)
  - Server invalidates all callbacks on write
  - Read, write are to local cached copy
  - Open to local cache; if callback not revoked
- Directory and symbolic links are also cached in later versions
- AFS uses UNIX calls for cached copies

Write sharing of two files

- Client1 opens file for writing
  - If no local copy, fetch a copy V1 from server
  - Modify local cached copy
- At this point, client2 opens file for writing
  - If no local copy, fetch a copy V1 from server
  - Modify local cached copy
- Client1 closes file (V2)
  - Server revokes call back on Client2; the next open will not use cached entry
  - Client1 replaces file version in server to V2
- Client 3 opens file for reading
  - Fetches a copy V2 from server
  - Read local copy
- Client 2 closes file (V3)
  - Server revokes call back on Client1 and Client3
  - Replaces file version to V4
- Client 3 reads local copy (V2)

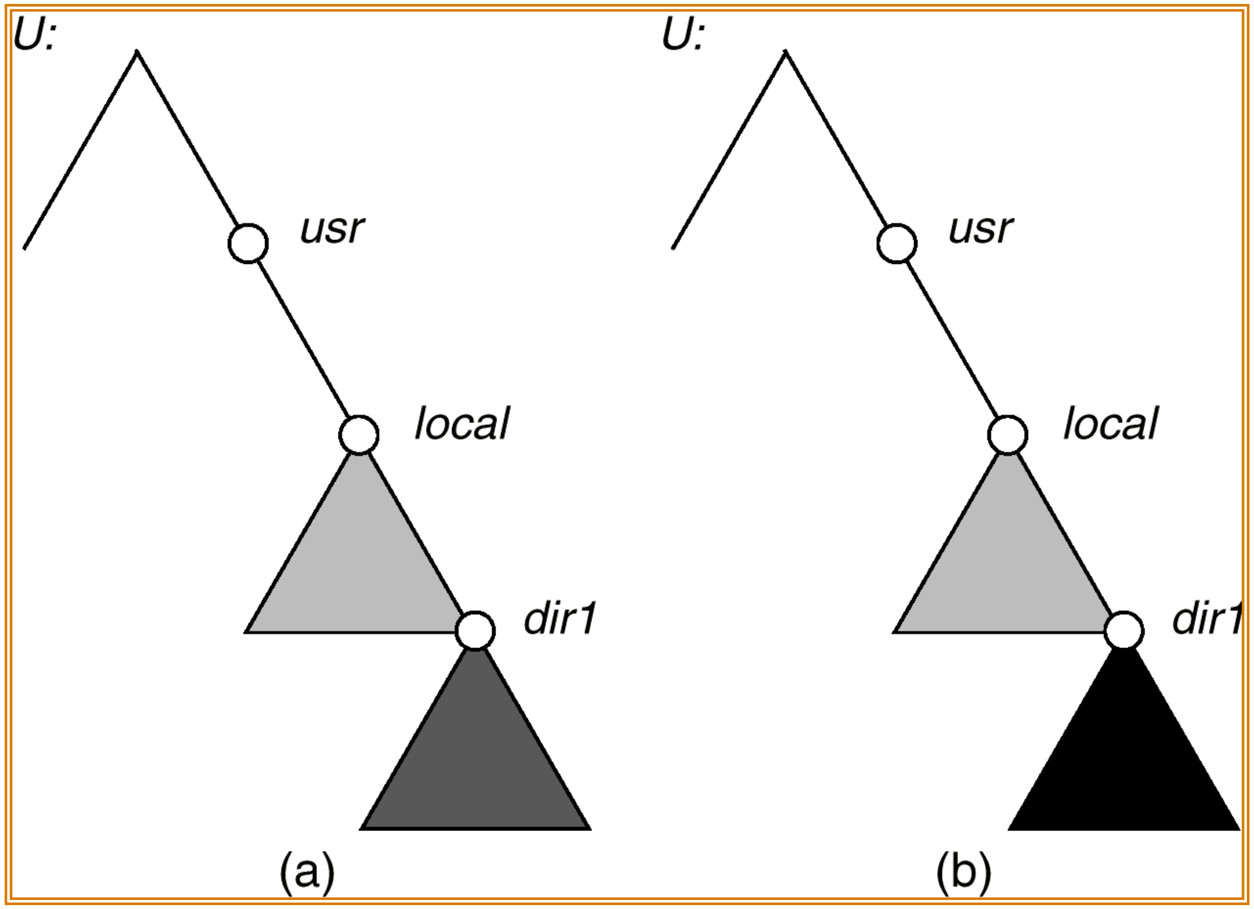
Design principles for AFS and Coda

- Workstations have cycles to burn - use them
- Cache whenever possible
- Exploit file usage properties
  - Temporary files are not stored in AFS
  - Systems files use read-only replication
  - Minimize system wide knowledge and change
  - Trust the fewest possible entities
  - Batch if possible

NFS

- Interconnected workstations viewed as a set of independent machines with independent file systems, which allows sharing among these file systems in a transparent manner.
  - A remote directory is mounted over a local file system directory. The mounted directory looks like an integral subtree of the local file system, replacing the subtree descending from the local directory.
  - Specification of the remote directory for the mount operation is nontransparent; the host name of the remote directory has to be provided. Files in the remote directory can then be accessed in a transparent manner.
  - Subject to access-rights accreditation, potentially any file system (or directory within a file system), can be mounted remotely on top of any local directory.

Mounting in NFS



Mounts

Cascading mounts

NFS Mount Protocol

- Establishes initial logical connection between server and client.
- Mount operation includes name of remote directory to be mounted and name of server machine storing it.
  - Mount request is mapped to corresponding RPC and forwarded to mount server running on server machine.
  - Export list – specifies local file systems that server exports for mounting, along with names of machines that are permitted to mount them.
- Following a mount request that conforms to its export list, the server returns a file handle—a key for further accesses.
- File handle – a file-system identifier, and an inode number to identify the mounted directory within the exported file system.
- The mount operation changes only the user's view and does not affect the server side.

NFS Protocol

- Provides a set of remote procedure calls for remote file operations. The procedures support the following operations:
  - searching for a file within a directory
  - reading a set of directory entries
  - manipulating links and directories
  - accessing file attributes
  - reading and writing files
- NFS servers are *stateless*; each request has to provide a full set of arguments.
- Modified data must be committed to the server’s disk (write through) before results are returned to the client (lose advantages of caching).
- The NFS protocol does not provide concurrency-control mechanisms.

NFS Path-Name Translation

- Performed by breaking the path into component names and performing a separate NFS lookup call for every pair of component name and directory vnode
  - /usr/homes/surendar/file.txt looks up /, /usr, /usr/homes ..
- To make lookup faster, a directory name lookup cache on the client's side holds the vnodes for remote directory names.

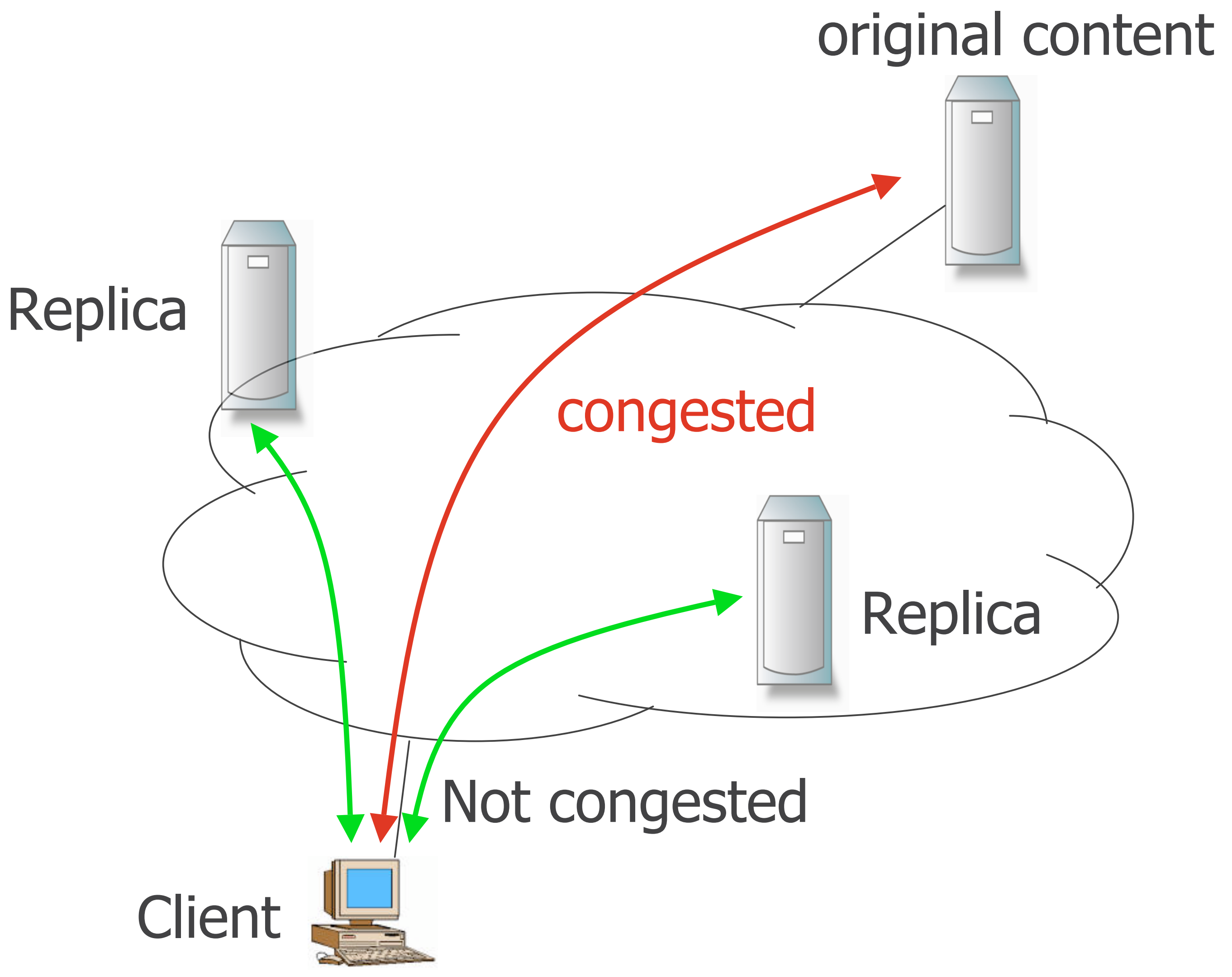
NFS Remote Operations

- Nearly one-to-one correspondence between regular UNIX system calls and the NFS protocol RPCs (except opening and closing files)
- NFS adheres to the remote-service paradigm, but employs buffering and caching techniques for performance
  - Stateless server
- File-blocks cache – when a file is opened, the kernel checks with the remote server whether to fetch or revalidate the cached attributes. Cached file blocks are used only if the corresponding cached attributes are up to date. Clients revalidate every so often (30-60 seconds)
- File-attribute cache – the attribute cache is updated whenever new attributes arrive from the server
- Clients do not free delayed-write blocks until the server confirms that the data have been written to disk

Extra material

- Oceanstore: An architecture for Global-Scale Persistent Storage – University of California, Berkeley. ASPLOS 2000
- Chord
- Content Distribution Network

Content Distribution Networks (slides courtesy Girish Borkar: Udel)



Persistent store

E.g. files (traditional operating systems), persistent objects (in a object based system)

- Applications operate on objects in persistent store
  - Powerpoint operates on a persistent .ppt file, mutating its contents
  - Palm calendar operates on my calendar which is replicated in myYahoo, Palm Desktop and the Pilot itself
- Storage is cheap but maintenance is not

  ~ 500 $/GB (Al Spector @ Transarc/IBM)

Global Persistent Store

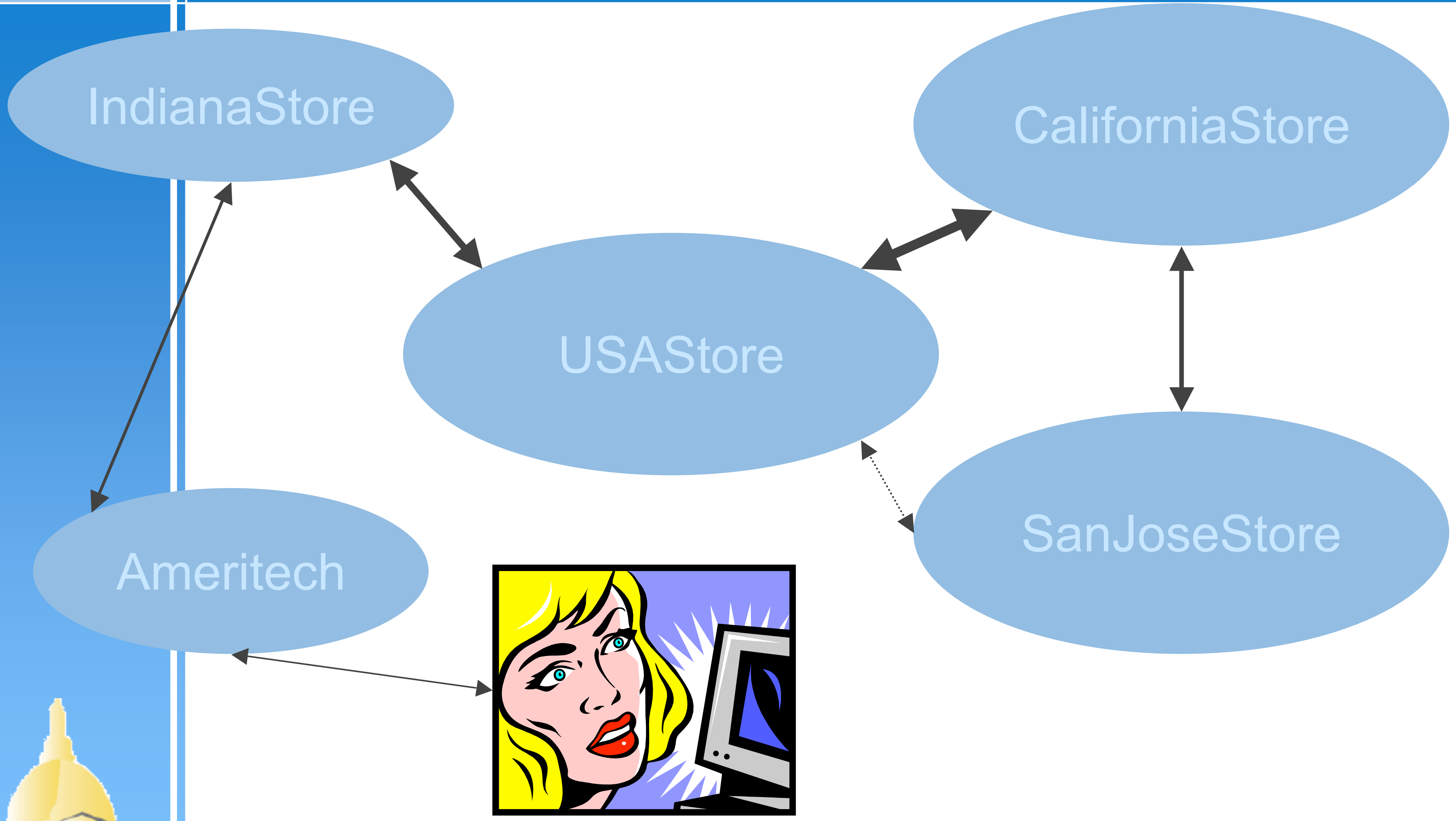
- Persistent store is fundamental for future ubiquitous computing because it allows "devices" to operate transparently, consistently and reliably on data.

- Transparent: Permits behavior to be independent of the device themselves
- Consistently: Allows users to safely access the same information from many different devices simultaneously.
- Reliably: Devices can be rebooted or replaced without losing vital configuration information

Persistent store on a wide-scale

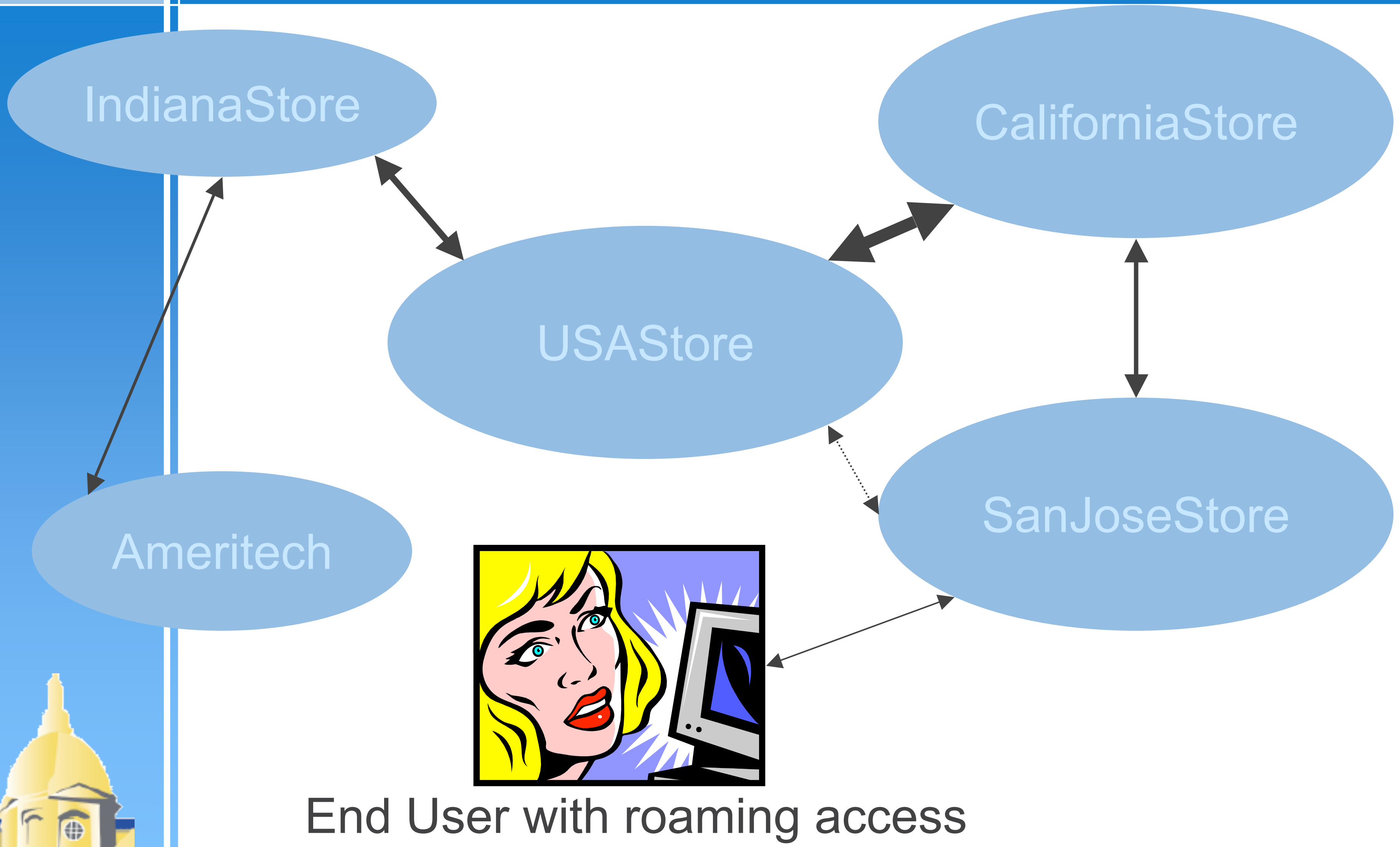
- 10 billion users, 10,000 files per user = 100 trillion files!!
- Information:
  - should be separated from location. To achieve uniform and highly-available access to information, servers must be geographically distributed, but exploit caching close to clients for performance
  - must be secure
  - must be durable
  - must be consistent

Oceanstore system model: Data Utility



End User with roaming access

Oceanstore system model: Data Utility



Oceanstore Goals

- Untrusted infrastructure (utility model – telephone)
  - Only clients can be trusted
  - Servers can crash, or leak information to third parties
  - Most of the servers are working correctly most of the time
  - Class of trusted servers that can carry out protocols on the clients behalf (financially liable for integrity of data)
- Nomadic Data Access
  - Data can be cached anywhere, anytime (promiscuous caching)
  - Continuous introspective monitoring to locate data close to the user

Oceanstore Persistent Object

- Named by a globally unique id (GUID)
- Such GUIDs are hard to use. If you are expecting 10 trillion files, your GUID will have to be a long (say 128 bit) ID rather than a simple name
  - passwd vs 12agfs237dfdfhj459uxzozfk459ldfnhgga
- self-certifying names
  1. secureHash(/id=surendar,ou=ND,key=<SecureKey>/etc/passwd) -> uniqueId
  2. Map uniqueId->GUID
  - Users would use symbolic links for easy usage
    - /etc/passwd -> uniqueId

SecureHash

- Pros:
  - The self-certifying name specifies my access rights
- Cons:
  - If I lose the key, the data is lost
    - Key management issues
      - Keys can be upgraded
      - Keys can be revoked
  - How do we share data?

Access Control

- All read-shared-users share an encryption key
  - Revocation:
    - Data should be deleted from all replicas
    - Data should be re-encrypted
    - New keys should be distributed
    - Clients can still access old data till it is deleted in all replicas
- All writes are signed
  - Validity checked by Access Control Lists (ACLs)
  - If A says trust B, B says trust C, C says trust D, what can you infer about A ? D

Oceanstore Persistent Object

- Objects are replicated on multiple servers. Replicated objects are not tied to particular servers i.e. floating replicas
- Replicas located by a probabilistic algorithm first before using a deterministic algorithm
- Data can be active or archival.
  - Archival data is read-only and spread over multiple servers – deep archival storage

Updates

- Objects are modified through updates (data is never overwritten) i.e. versioning system
- Application level conflict resolution
- Updates consist of a predicate and value pair. If a predicate evaluates to true, the corresponding value is applied.
  1. <room 453 free?>, <reserve room>
  2. <room 527 free?>, <reserve room>
  3. <else> <go to Jittery Joes>
- This is similar to Bayou

Introspection

- Oceanstore uses introspection to monitor system behavior
- Use this information for cluster recognition
- Use this information for replica management