CSci 5105

Introduction to Distributed Systems

Distributed File Systems: AFS and Coda

AFS

<u>Andew File System</u> (AFS)

- Scalable (100's - 1000's of workstations) - unlike NFS
- Transparent access to remote files
- Maintain Unix file interface

<u>Features</u>

- Whole-file serving
- Whole-file caching

Now: open-sourced **OpenAFS**

AFS (cont'd)

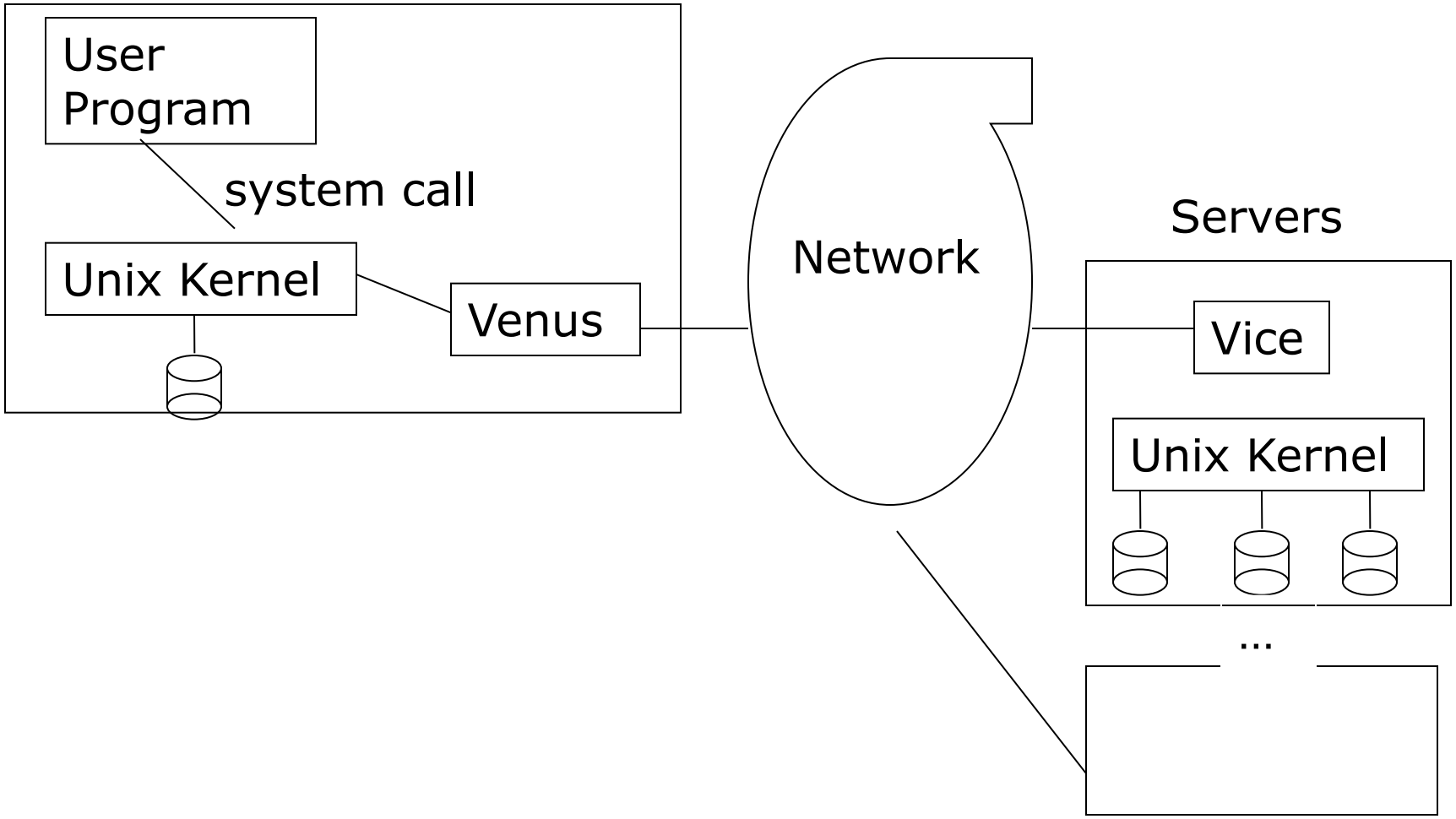
- Session semantics
  - client open => entire file is fetched from server and stored/cached on client disk
  - client reads/writes are done locally
  - client close => file sent back to server, but also kept locally
  - If multiple writers, last update "wins"

AFS (cont'd)

- AFS design guided by file usage observations:
  - most file are read-only, infrequently updated, small, limited sharing=>client caching may work well
  - programs access only a few files (and files are small) => client can "cache" (on local disk) their "working set" of files
  - file access locality (entire file is accessed sequentially) => whole file caching is reasonable

AFS (cont'd)

- AFS architecture



AFS (cont'd)

- Servers are dedicated to file management - they run only Vice server
  - Files are either <u>local</u> or <u>shared</u> (remote) [stored on server, cached locally]

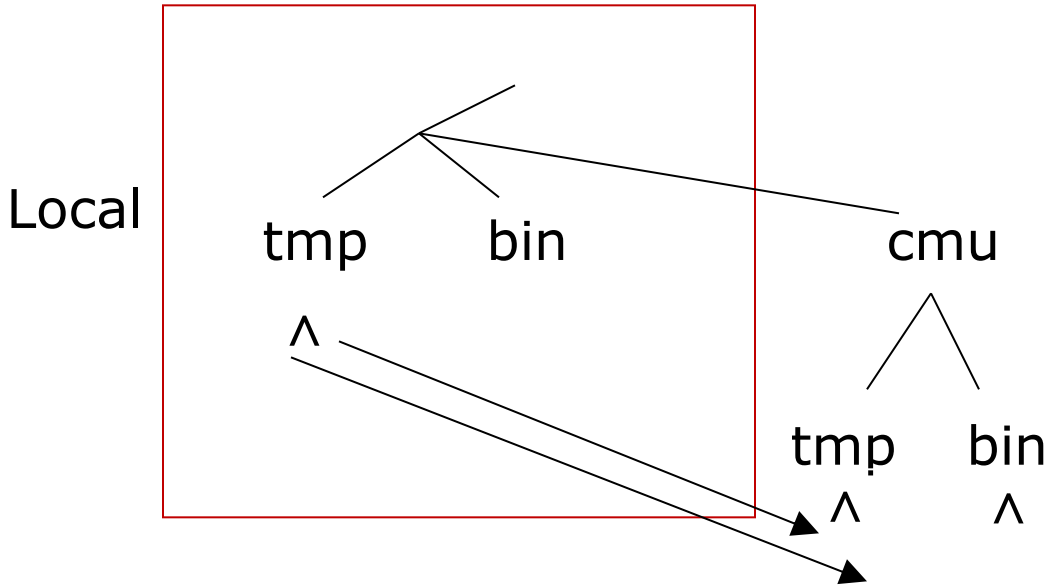
- <u>Venus</u> (client-side user-space program)
  - manages client cache for shared files (partition of local disk)
  - uses LRU replacement when local disk is full
  - kernel is modified to handle Venus interface

AFS (cont'd)

- Vice (server-side user-space program)
  - keeps track of clients that have cached a copy of all files
  - stateful: state is updated atomically to disk for fault tolerance

AFS (cont'd)

- Unlike mounting, local vs shared is less transparent
  - most files are shared (e.g. user directories) to allow them to be accessed from any workstation



Symbolic links: gain some transparency back

AFS (cont'd)

- File are grouped in <u>volumes</u> (collection of directories)
  - read only
  - read/write
  - each file is identified by a unique 96 bit file id (fid)

- Volumes may be migrated/replicated

AFS (cont'd)

- Cache Coherence
  - Vice keeps track of client copies so that client can be notified if update occurs
- Callbacks
  - when a client closes, Vice gets the updated file (if client updated the file)
  - Vice notifies the Venus processes that it knows contain copies

AFS (cont'd)

- Each Venus marks the cached copy as cancelled
  - When each Venus tries to access the file it will be invalid (cancelled) and the file will be re-fetched
  - client can indicate it doesn't care to be notified – efficiency'
  - callbacks must be renewed after a time T (mins)

    since cancel messages may have been lost

AFS (cont'd)

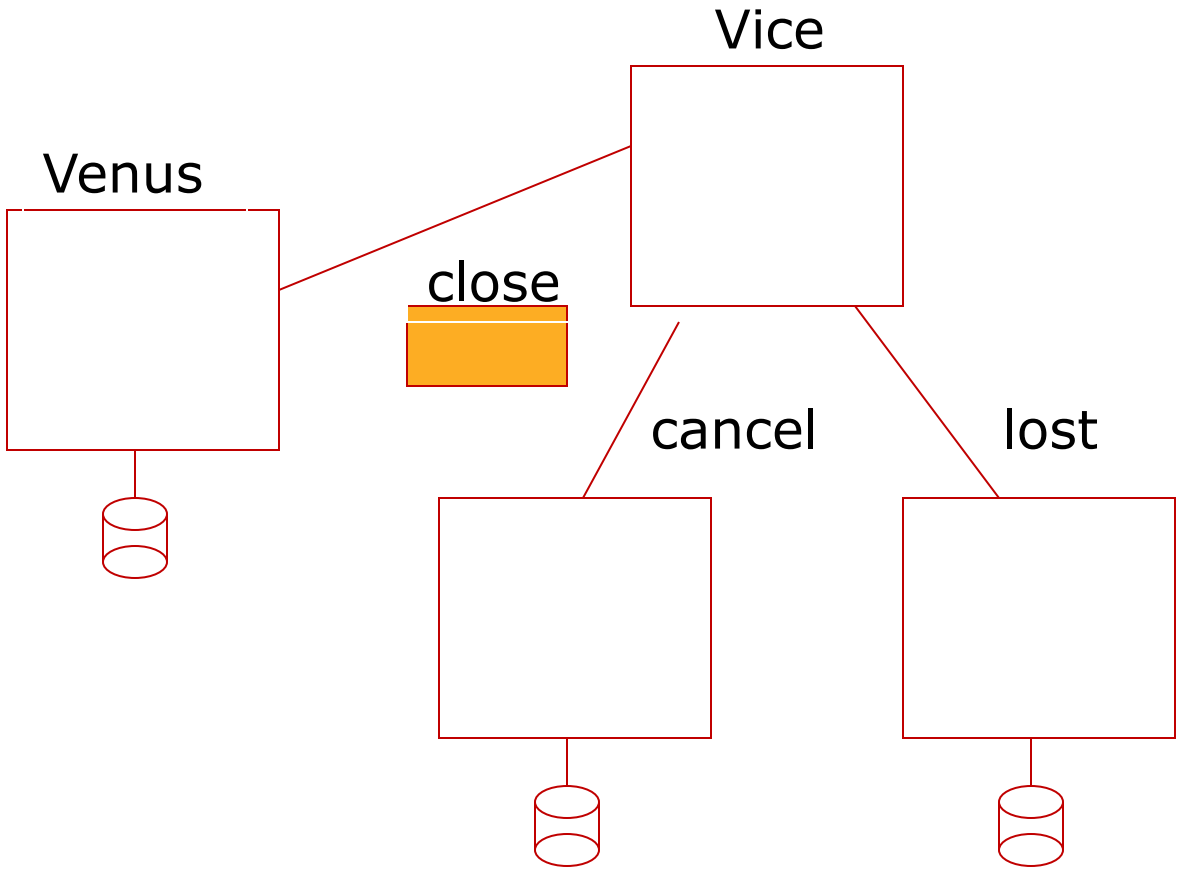
- after T, Venus asks Vice for a time-stamp on latest update to see if still valid
- after a client reboot, Venus asks Vice for a time-stamp on latest update for all cached files to see if still valid

AFS (cont'd)

- after T, Venus asks Vice for a time-stamp on latest update to see if still valid

- after a client reboot, Venus asks Vice for a time-stamp on latest update for all cached files to see if still valid

AFS (cont'd)



AFS (cont'd)

- <u>AFS vs. NFS</u>
  - Much less client/server communication in AFS
  - NFS every open, access of a new file block, invalidate every 3 sec, flush every 30 sec
  - AFS callbacks issued from the <u>server</u> only when a file is updated (should be infrequent based on earlier observations)

AFS (cont'd)

- <u>AFS vs. NFS</u>
  - Time-out vs. Callback
  - AFS provides a well-defined approximation to one-copy semantics via callback (not feasible to propagate all writes to all clients)
  - AFS may get stale files

    client issues an open but may get an old copy if cached previously (if callback message is lost) : at most T minutes out of date

Coda

- Frequent data unavailability in AFS
- Server Replication
  - More difficult to get partitioned from all servers
  - Consistency?

- Disconnected Operation
  - If no available servers, attempt to work off of the local cache
  - Consistency?

CODA

- <u>Coda Features</u>
  - Based on AFS (~ Unix semantics)
  - Replication of read-only volumes limits performance
  - E.g. bboards, databases
  - Fault tolerance

In AFS servers may be unavailable for stretches of time

- even if file cached locally, will be invalidated

CODA (cont'd)

- Support for portable computers and wireless networks
  - Files available to client even if disconnected from network

CODA (cont'd)

- <u>Goals</u>
  - Constant data availability
  - Replicate read-write volumes
  - Allow clients to continue processing on local files (if disconnected)

- VSG - volume storage group (set of Vice servers that replicate a volume)

- AVSG - available servers w/r to a client in VSG, if |AVSG|=0, client is disconnected

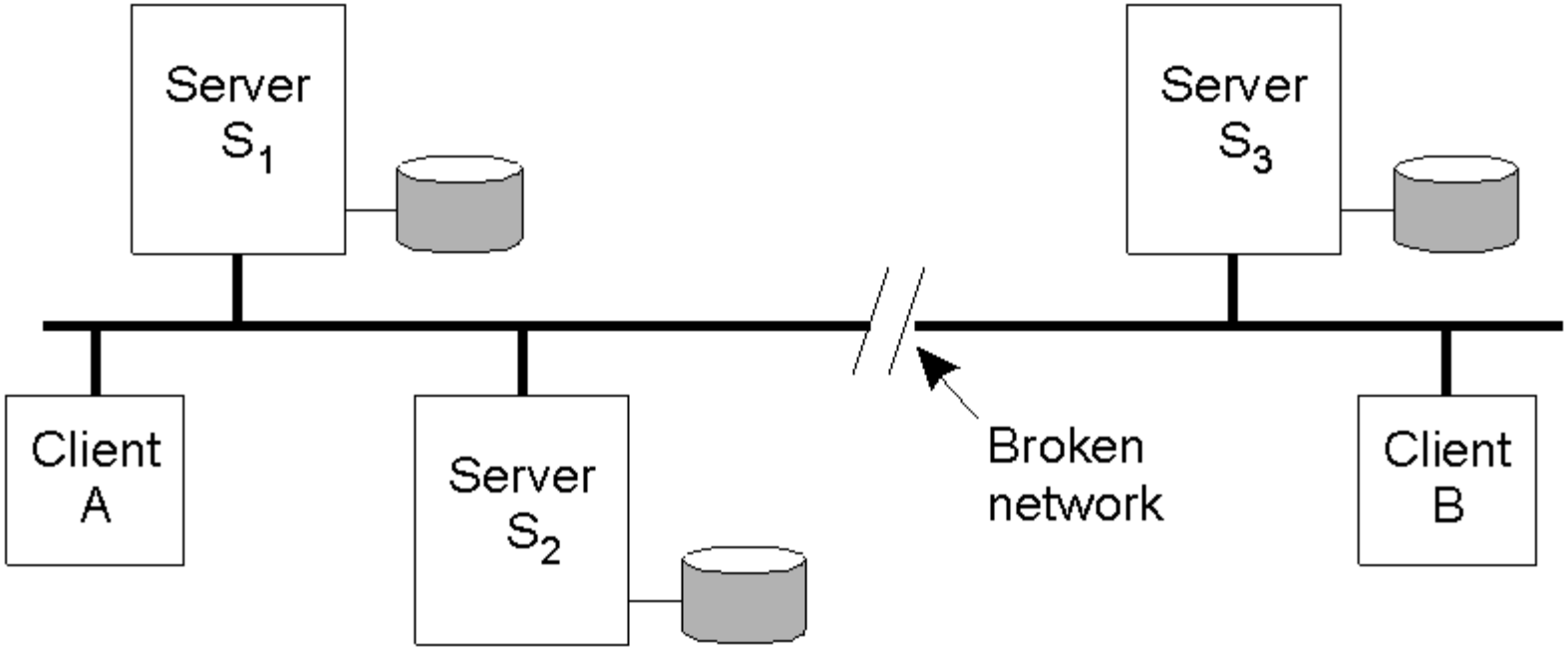
Server Replication

- ***read-one, write-all approach***
- Each client has a ***preferred server***
  - Holds all callbacks for client
  - Answers all read requests from client

- Latest version
  - sole owner: no callbacks are used

Server Replication

- Two clients with different AVSG for the same replicated file.



CODA (cont'd)

- When Venus <u>opens</u> a file, it asks of the AVSG servers to do it (preferred server handles callbacks)

- When Venus <u>closes</u> (updates) a file, it broadcasts the update to all servers in AVSG

CODA (cont'd)

- Replication on read/write volumes
  - Introduces the problem of replica coherence
    - not the problem of multiple different Venus copies
    - problem of multiple different server (Vice) copies
  - CVV Coda Version Vector (~ vector clock)
  - one entry for each server that stores a replica
  - each entry is an estimate of the # of mods made to that copy
  - Resolve inconsistencies using CVV's

CODA (cont'd)

- <u>Accessing replicas</u>
  - Read-one

open: get a copy from preferred server in AVSG

  - Write-all

close: all copies are written via multicast

- if a server is unavailable, mark copy to propagate later when server becomes reachable

CODA (cont'd)

- <u>Disconnection</u>
  - Before disconnection …

Coda allows users to specify a prioritized list of files/directories that Venus should keep cached (sticky)

{similar to disabling callbacks in AFS}

CODA (cont'd)

- After re-connection
  - Re-integration
  - Local cached copies that were updated are compared with latest copies at AVSG
    - compare CVVs
    - if one replica only, easy (may be the case) - server copy dominates
    - server copies have priority over cache copies
    - may need manual intervention by the file owner if conflicts {as before}

CODA (cont'd)

- <u>Coda Advantages</u>
  - Higher availability

  |AVSG| normally > 0, R-W replicas

  allow clients to operate when disconnected

- Performance issues
  - all replicated servers can serve a file (load sharing)
  - multicast and coherence protocol creates performance degradation

Next Time

- Case Studies
  - Grid, P2P
- Read papers on website

- Have a great weekend!