IF3230 – Sistem Terdistribusi

Distributed File Systems – Andrew File Systems

Achmad Imam Kistijantoro (imam@informatika.org)

Judhi Santoso (judhi@informatika.org)

Anggrahita Bayu Sasmita (bayu.anggrahita@informatika.org)

Topik

- Generic Distributed File System
- NFS Network File System
- **AFS Andrew File System**
- Google File System
- Hadoop Distributed File System (HDFS)

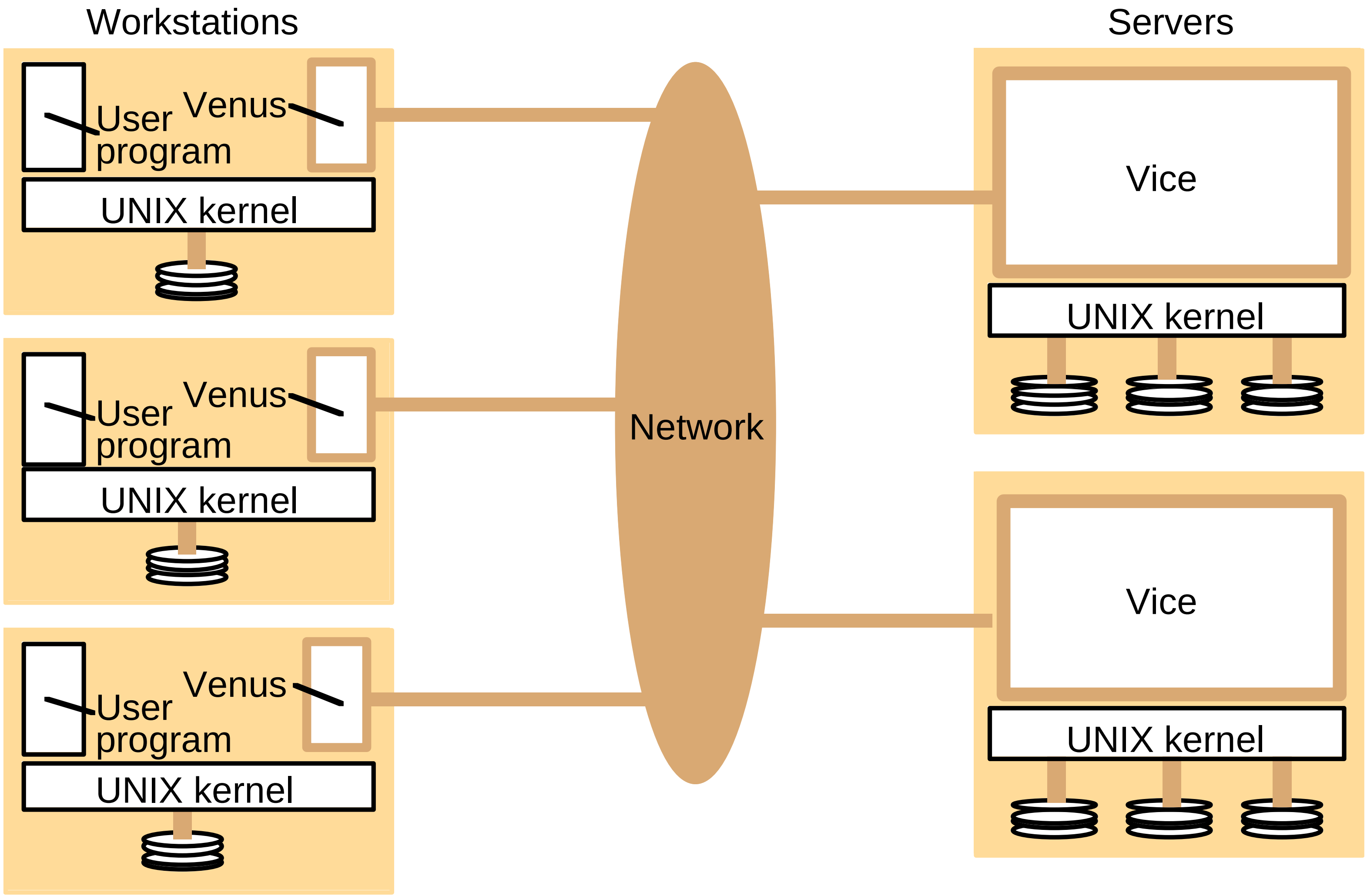
Andrew File System

- berbeda dengan NFS, AFS lebih mengutamakan scalability.
- scalability dicapai dengan melakukan caching keseluruhan file pada client nodes.
- Sebuah client cache dapat menyimpan beberapa ratus file yang most recently used pada komputer tersebut
- Cache bersifat permanent.
- Saat client membuka file, cache diperiksa dan digunakan jika sudah pernah di cache sebelumnya.

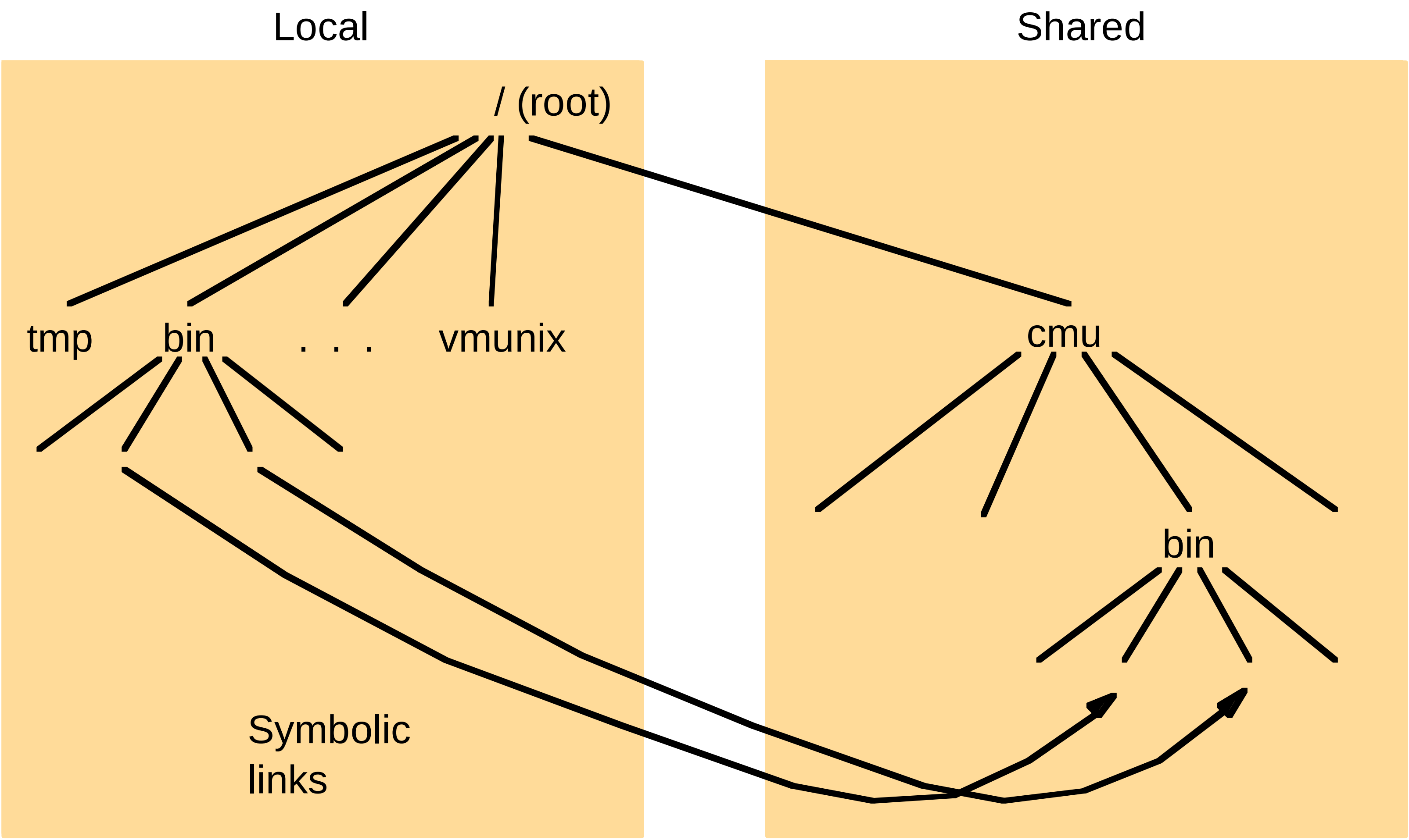
Andrew File System - Typical Scenario

- Saat code client membuka file, client cache dibaca terlebih dahulu. Jika tidak ada, lokasi server ditentukan dan file dibaca dari server tersebut.
- Copy file disimpan pada sisi client dan dibuka.
- reads dan writes berikutnya dilakukan pada copy client.
- Saat client menutup file – jika file berubah, file tersebut dikirim balik ke server. Copy file tetap disimpan pada client.
- Contoh: library dan command files. File yang hanya digunakan oleh single user => mayoritas akses file pada Unix
- Principle: Make the common case fast.

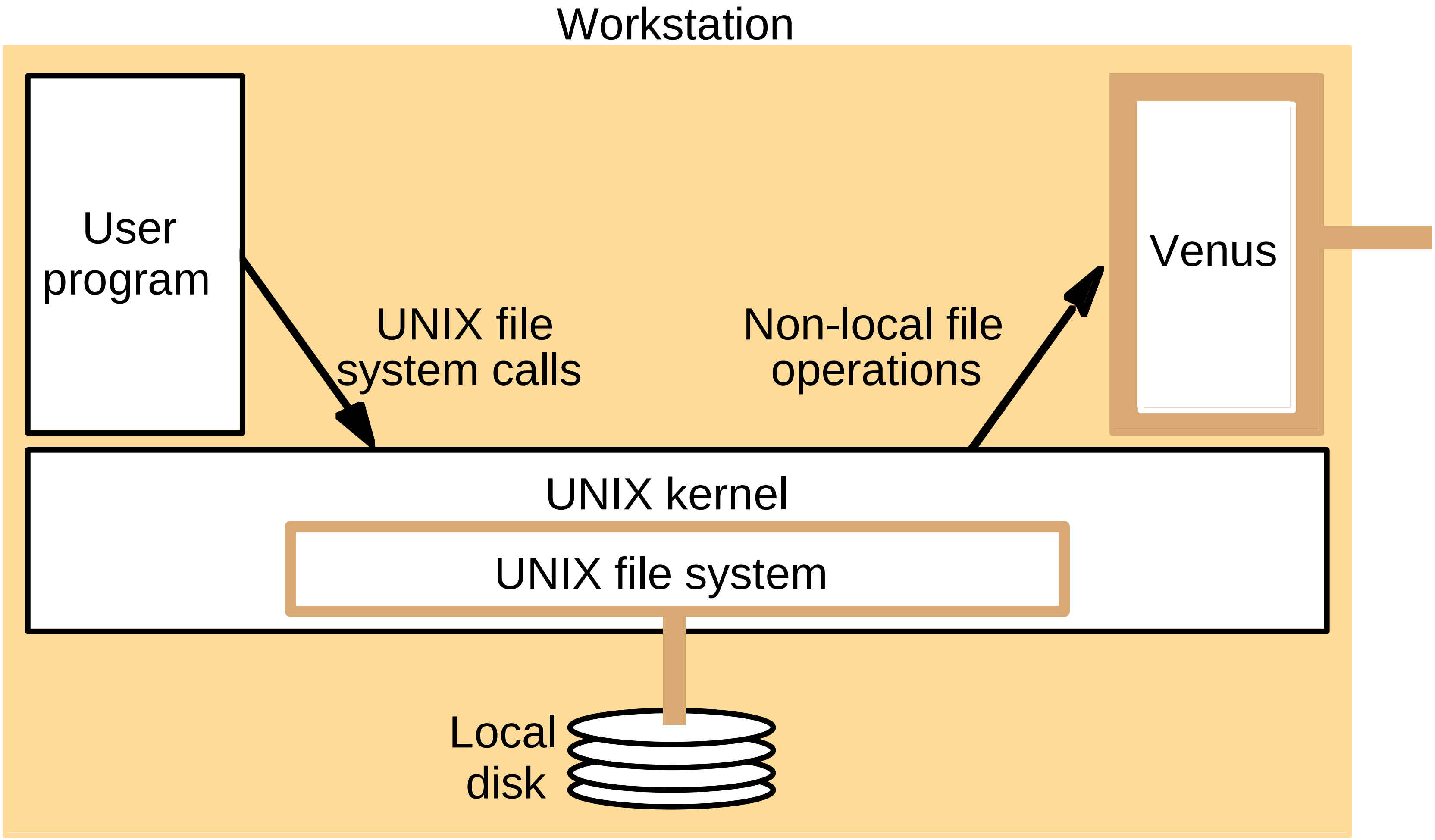
Distribution of processes in the Andrew File System



File name space seen by clients of AFS



System call interception in AFS



Implementation of file system calls in AFS

User process	*UNIX kernel*	*Venus*	*Net*	*Vice*
open(FileName, mode)	If *FileName* refers to a file in shared file space, pass the request to Venus. Open the local file and return the file descriptor to the application.	Check list of files in local cache. If not present or there is no valid *callback promise*, send a request for the file to the Vice server that is custodian of the volume containing the file. Place the copy of the file in the local file system, enter its local name in the local cache list and return the local name to UNIX.	→ ←	Transfer a copy of the file and a *callback promise* to the workstation. Log the callback promise.
read(FileDescriptor, Buffer, length)	Perform a normal UNIX read operation on the local copy.			
write(FileDescriptor, Buffer, length)	Perform a normal UNIX write operation on the local copy.			
close(FileDescriptor)	Close the local copy and notify Venus that the file has been closed.	If the local copy has been changed, send a copy to the Vice server that is the custodian of the file.	→	Replace the file contents and send a *callback* to all other clients holding *callback promises* on the file.

The main components of the Vice service interface

Fetch(fid) -> attr, data	Returns the attributes (status) and, optionally, the contents of file identified by the *fid* and records a callback promise on it.
Store(fid, attr, data)	Updates the attributes and (optionally) the contents of a specified file.
Create() -> fid	Creates a new file and records a callback promise on it.
Remove(fid)	Deletes the specified file.
SetLock(fid, mode)	Sets a lock on the specified file or directory. The mode of the lock may be shared or exclusive. Locks that are not removed expire after 30 minutes.
ReleaseLock(fid)	Unlocks the specified file or directory.
RemoveCallback(fid)	Informs server that a Venus process has flushed a file from its cache.
BreakCallback(fid)	This call is made by a Vice server to a Venus process. It cancels the callback promise on the relevant file.