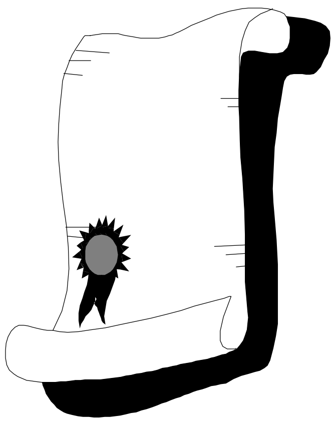
Moving UNIX Client/Server Applications to Production

Session 350

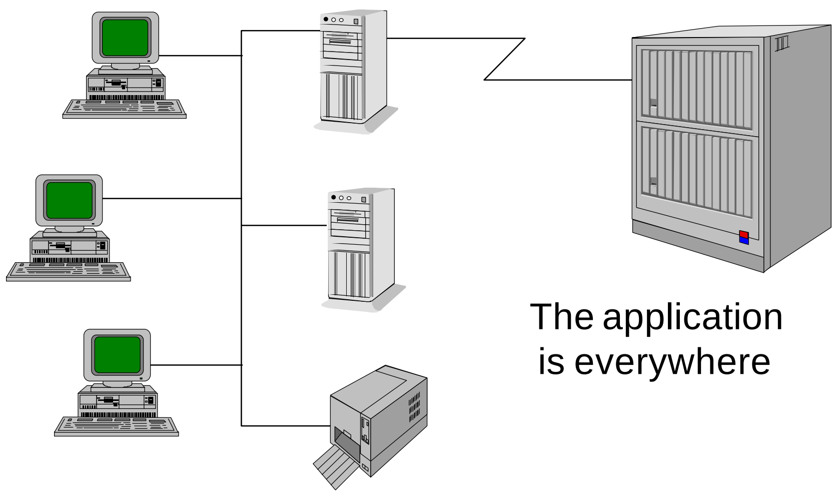
Michelle De Hertogh
Texas Instruments

Agenda



- Client/Server Topology
- Composer C/S application
- UNIX server environment
- Build versus runtime on the UNIX platform
- UNIX runtime necessary
- Production Plans for runtime

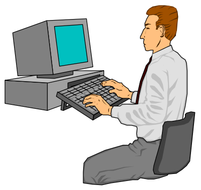
Client/Server Topology



Application Components

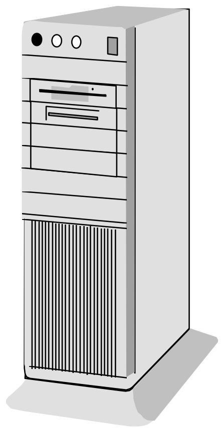
- Any client/server application has multiple layers
  - – Workstation layer
  - – Communication layer
  - – Server layer
- Each layer can contain pieces of the application

Workstation Layer



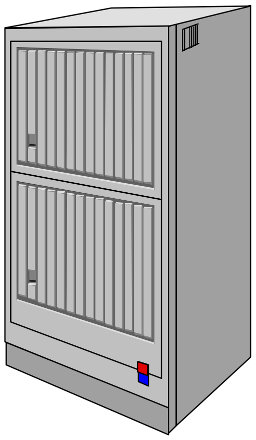
- Provides users access to the applications
- Executes the client portion of the application
- Provides access to the network
- Provides a local hard drive for storage of applications and data

Communication Layer



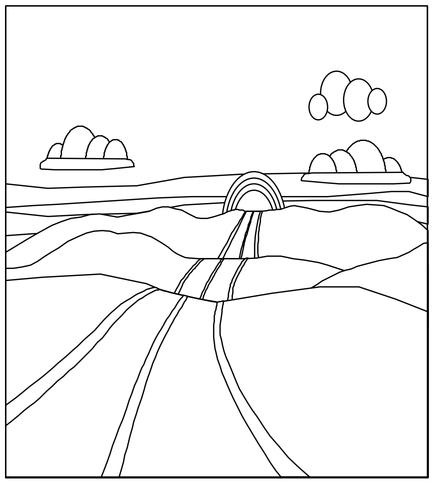
- Enables the communication from the workstation to the server layer
- Provides file servers for storage of common programs and data
- Stores client application load modules

Server Layer



- Executes the server portion of the application
- Holds the central database
- Provides data to the workstation layer

Composer Applications



- Workstation
  - – Client applications
  - – GUI runtime
  - – Client manager
- UNIX server(s)
  - – Server applications
  - – UNIX runtime

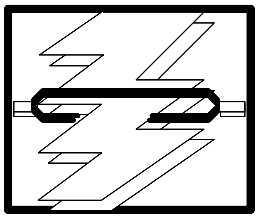
Composer Client Applications

- Include generated executables and DLLs
- Package multiple procedure steps in client load modules
- Can be located on a file server
- Ensure that related executables can be accessed
- Locate DLL and the help files for the load module in the same directory as the executable

Composer GUI Runtime

- Composer requires three runtime DLL files
- Files can be located within the application directory or in a workstation’s path statement
- Runtime DLL’s can be located on a file server
- Runtime DLL’s are release-specific (for example, Composer applications require Composer runtimes)

Composer Client Manager



- Every workstation must have the Client Manager executing
- Each workstation must have a unique machine name
- It can physically reside on the file server or user's workstation

UNIX Server Environment

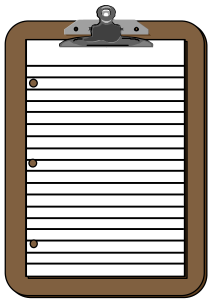
C/S Build Environment

- Implementation toolset
- Target configuration(s)
- Builds inqload directory(s)
- Builds aeenv file(s)
- Builds load module exe(s)
- Uses environment variables
  - AEHOME/AEPATH
  - DBMS-specific
  - IEFH
- Log files (aef)

C/S Runtime Environment (Transaction Enabler)

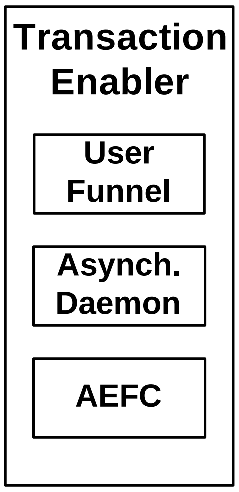
- TE uses inqload directory(s)
- TE uses aeenv file(s)
- TE Executes load modules
- Uses Environment variables
  - AEHOME/AEPATH
  - DBMS-specific
- Log files (AD,UF,AEFC)
- User exits, Security, AEFC
- Shell Scripts

Assumptions



- Application components and environment are selected
- UNIX server is selected
- A client platform is selected
- A communications protocol is selected
- Distributed Processing is the client/server style selected

Transaction Enabler is Required

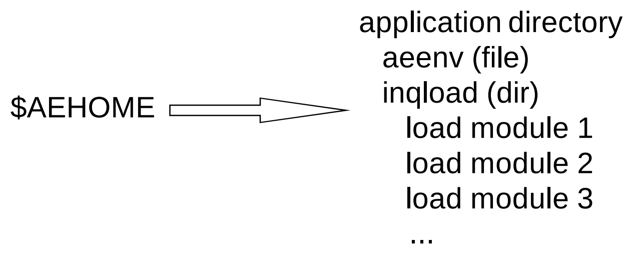


- Transaction Enabler acts as the teleprocessing monitor for the UNIX platform
- Consists of three UNIX processes:
  - – Asynchronous Daemon (aefad)
  - – User Funnel (aefuf)
  - – Application Execution Facility Client (aefc)
- Need all three in production

Transaction Enabler

- Need one asynchronous daemon for executables to be:
  - loaded in memory
  - kept resident in memory
  - kept connected to the DBMS
  - kept shareable
- Need user funnels to:
  - allow multiple users to share a single aefad environment
  - connect GUI client managers to a UNIX server
- Need an aefc to monitor and dynamically change (if necessary) aefad

Required Directory/File Structure

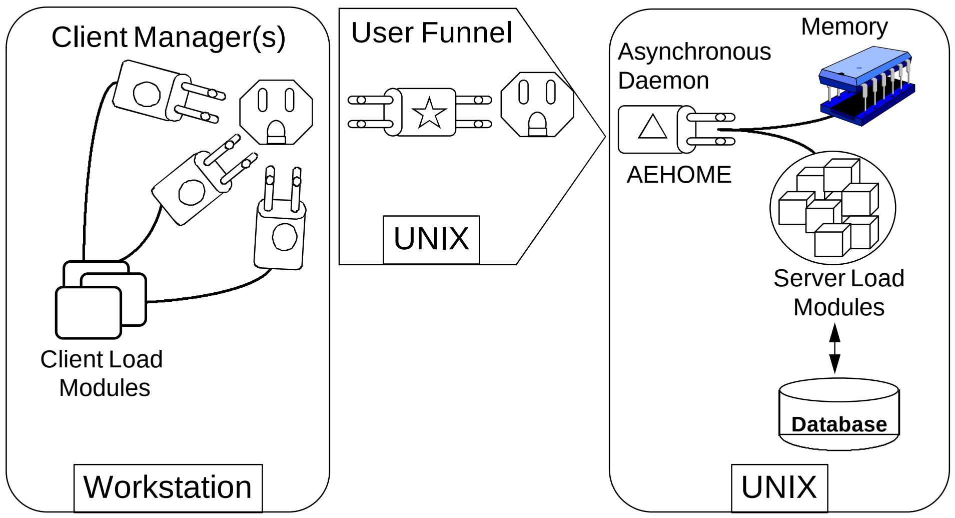


- Must have a directory named inqload that contains the load modules (executables)
- Must have a file at the inqload directory level named aeenv that contains the tran codes in the executables
- $AEHOME must point to the inqload directory

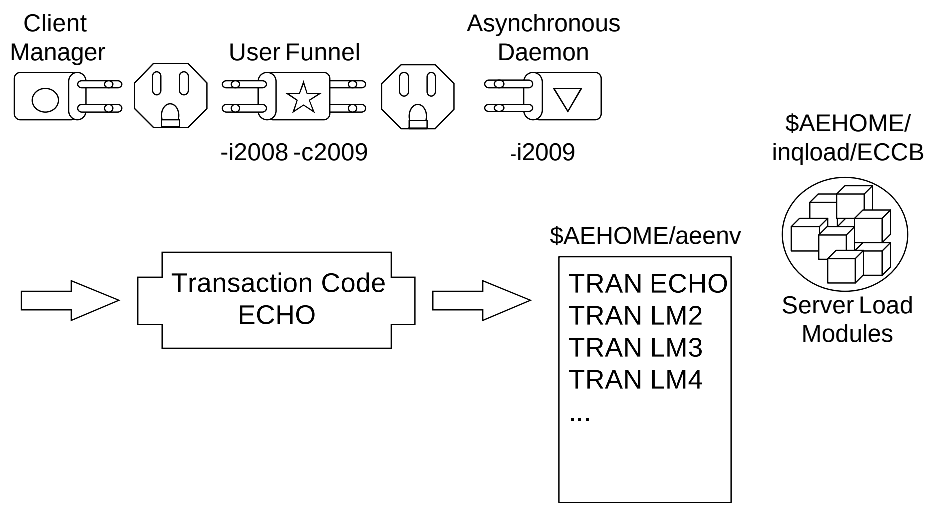
Optional Directory/File Structure

- aeenv file and inqload directory can have any parent directory
- Directory for runtime processes (aefad, aefuf, aefc)
- Directory for shell scripts
- Directory for log files

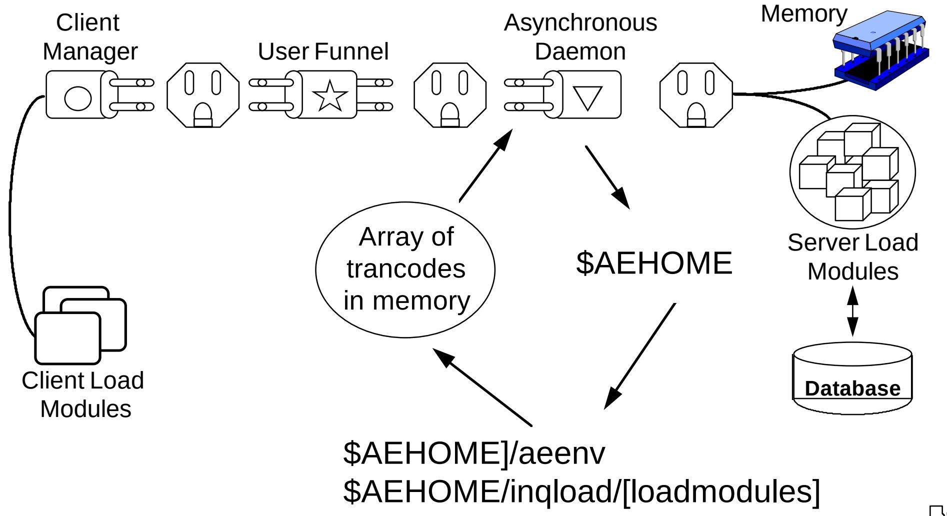
C/S Application Execution



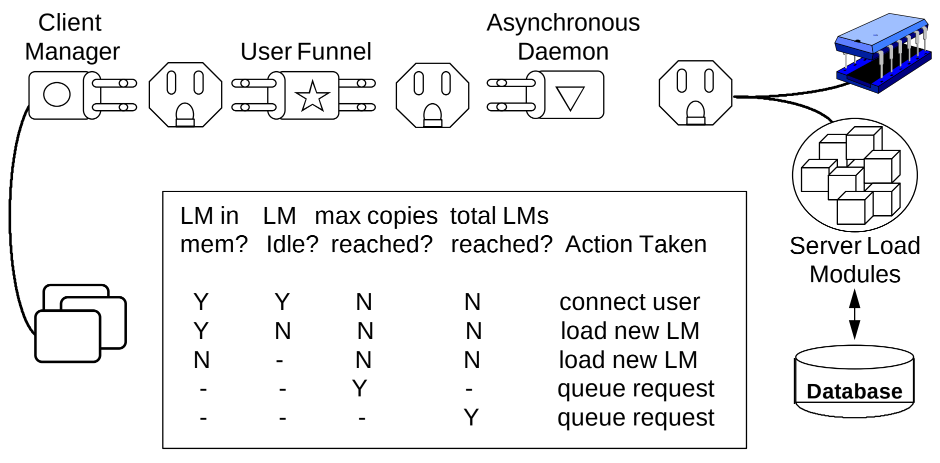
Single Transaction Execution



Asynchronous Daemon–Load Time

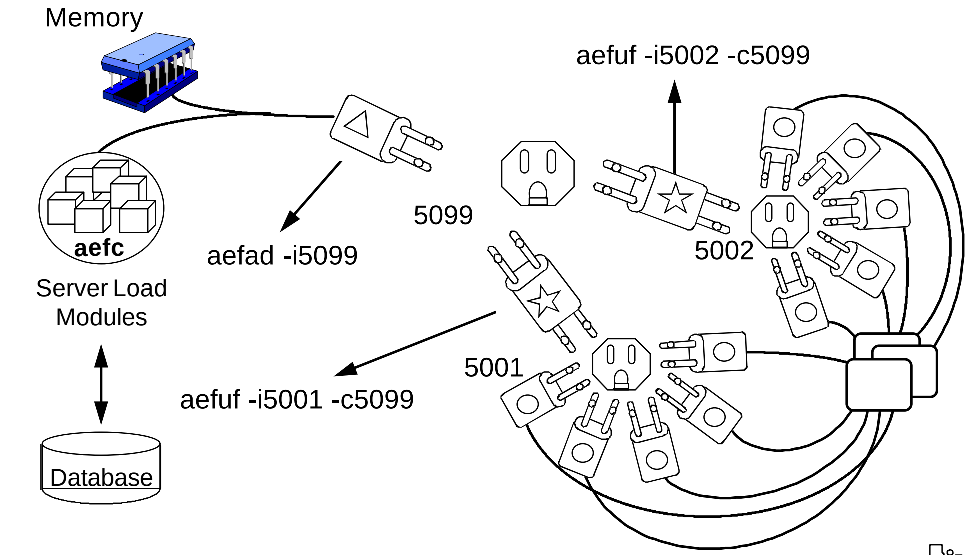


Asynchronous Daemon–Runtime

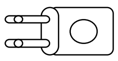
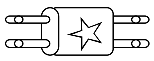
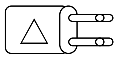


LM in mem?	LM Idle?	max copies reached?	total LMs reached?	Action Taken
Y	Y	N	N	connect user
Y	N	N	N	load new LM
N	-	N	N	load new LM
-	-	Y	-	queue request
-	-	-	Y	queue request

Scaleability

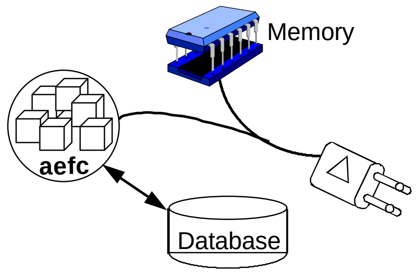


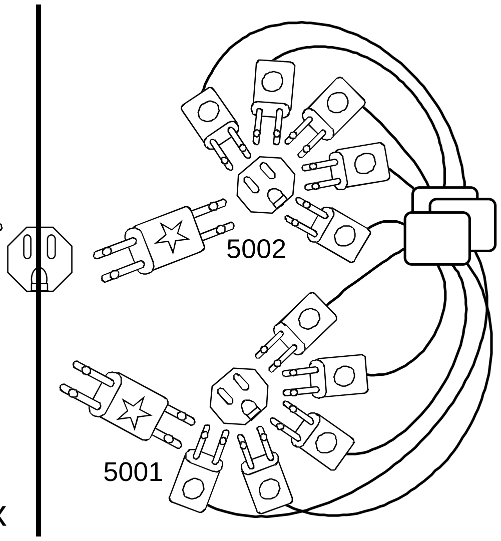
Critical Mass

Client Manager		Multiple Requests by Client LMs
User Funnel		~250 Client Requests per Funnel
Asynchronous Daemon		~UNIX Capacity, Default is -u512

- Divide # users by User Funnel capacity
- Determine user capacity of UNIX CPU
- Determine if multiple Daemons are necessary

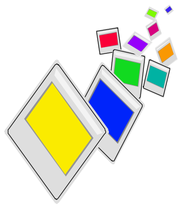
Separating Funnels from Daemons





- Increased TCP traffic pulls CPU cycles from application
- Move heavy TCP traffic to dedicated box

Production Plans for TE



- Asynchronous Daemon
  - Must be tuned for application
  - Only need one unless very large application
  - Use AEHOME instead of AEPATH
  - Only need single aeenv file
  - Only need single inqload directory
  - Must be assigned a unique port number

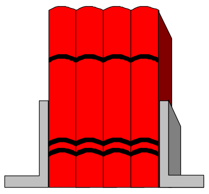
Production Plans for TE

- User Funnel
  - Determine how many
  - Determine what platform to run on
  - Each must be assigned a unique port number
- AEFC
  - One per daemon

Environment Variables

- Required
  - AEHOME (not AEPATH)
  - DBMS Requirements
- Optional
  - Runtime included in PATH
  - Shell script directory
  - Log file directory

Log Files



- AEFAD
  - aestats – application execution statistics
  - lgxxxxxx (pid #) – daemon process log file (leave off unless needed)
- AEFUF
  - lgxxxxxx (pid #) – user funnel process log file (leave off unless needed)
- AEFC
  - lgxxxxxx (pid #) – aefc process log file (leave off unless needed)

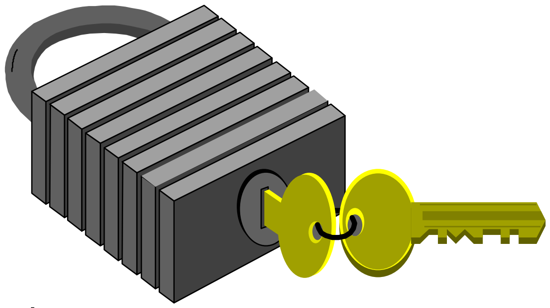
Shell Scripts

- Startup asynchronous daemon
- Startup user funnel(s) Pre-load load modules into memory
- Pre-load load modules into memory
- “Batch” scripts

User Exits

- tir?conn - database logon
- aefsecex - aefad security
- tirsecr - load module application security

Security



- aeenv file
- inqload directory
- load module execution
- aefc file
- aefad file
- aefuf file

What is Necessary for Production?

- Runtime (aefad, aefuf, aefc)
- Load module executables in inqload directory
- aeenv file
- Directory structure
- Environment variables
- Shell scripts
- User exits
- Security
- Log files

Moving UNIX Client/Server Applications to Production

Session 350

Michelle De Hertogh
Texas Instruments