NAME CACHES AND LDAP

- K.SRIRAGHAV – 3rd year CSE

Need for name caches

- In UNIX systems, the system call frequency for name-mapping operations (open, stat, lstat) constitute over 50% of the file system calls.
- Researchers also observed that in a large distributed system, a substantial portion of the network traffic is naming related.
- Hence there is a need for client to cache the result of a name resolution operation for a while, rather than repeating it every time the value is needed.

What is name cache?

- Caching name servers (*DNS caches*) store DNS query results for a period of time determined in the configuration (time-to-live) of each domain-name record.
- **Advantages:**
- Name caches improve the efficiency of the DNS by **reducing DNS traffic** across the distributed system
- Reducing load on authoritative name-servers, particularly root name-servers.
- Because they can answer questions more quickly, they also increase the performance of end-user applications that use the DNS.

Characteristics of name caches

- **High degree of locality of name lookup:**
  - The property of “locality of reference” has been observed in program execution , database access and file access.
  - Due to this locality feature, a reasonable size name cache , used for caching recently used information, can provide excellent hit ratios.
- **Slow update of name information database:**
  - Naming data has a high read-to-modify ratio.
  - This behaviour implies that the cost of maintaining the consistency of cached data is significantly low.

Characteristics of name caches - [contd]

- **On-use consistency of cached information:**
  - Name cache consistency can be maintained by detecting and discarding stale cache entries on use.
  - With On-use consistency checking, there is no need to invalidate all related cache entries when a naming data update occurs .

Types of name caches

- **Directory cache:**
  - Each entry consists of a directory page.
  - Used in systems that use the iterative method of name resolution.
  - All recently used directory pages that are brought to the client node during name resolution are cached for a while.
  - Ex : LOCUS
  - **Disadv:** For only one useful entry of a directory, an entire page of a directory blocks a large area of precious cache space.

Types of name caches-[contd]

- **Prefix cache:**
  - Used in naming systems that use the zone-based context distribution mechanism.
  - Each entry consists of a name prefix and the corresponding zone identifier.
  - Ex: Sprite , V-System
  - **Disadv** - Not suitable to use with structure-free context distribution approach.

Types of name caches - [contd]

- **Full-name cache:**
  - Each entry consists of an object's full pathname and the identifier and location of its authoritative name server.
  - So, requests for accessing an object whose name is available in local cache can be directly sent to authoritative name server
  - Ex : Galaxy
  - **Adv** - Used with any naming system
  - **Disadv** - Larger in size than prefix name caches.

Approaches for name cache implementation

- **Cache per process:**
  - Separate name cache is maintained for each process.
  - Each cache is maintained in the corresponding process's address space.
  - Therefore, accessing of cached information is fast and no memory area of the OS is occupied by the name caches.
  - Very process must create its own name cache from scratch.
  - Hence if the processes are short-lived, caches will have short lifetimes.
  - Hit ratio will be less for process-oriented caches due to start-up misses.
  - Ex. Sprite and V-System

Approaches for name cache implementation - s[contd]

- **A single cache for all processes of a node**
  - In this approach, a single name cache is maintained at each node for all the processes of that node.
  - When compared to the process-oriented name caches, these caches are larger in size.
  - Accessing of information is slow
  - However, cached information in a single name cache is long lived – therefore no start-up- misses.
  - So higher average hit ratio as compared to process-oriented caches.
  - Ex : Locus and Galaxy

Multicache consistency

- When a data update , related name cache entries become stale and must be invalidated or updated properly.
- Two methods for Multicache consistency are :
  - Immediate invalidate
  - On-use update
- **On-use update :**
  - Most commonly used method for maintaining name consistency.
  - No attempt is made to invalidate all related cache entries when a naming update occurs.
  - When a client uses a stale cached data, it is informed by the naming system that data being used is either incorrectly specified.
  - On receiving a negative reply, necessary steps are taken to obtain the updated data.

Immediate invalidate

First approach

- Whenever a naming update operation is done, an invalidate message identifying the data to be invalidated is broadcast to all the nodes in the system
- Each node's name cache are then examined for the presence of updated data
- If it is present, the corresponding cache entry is invalidated.

Second approach

- The storage node of naming data keeps a list of nodes on which the data is cached.
- When a storage node receives a request for a naming data update, only the nodes in the corresponding list are notified to invalidate.

LDAP

Lightweight Directory Access Protocol

What is LDAP ? What is directory?

- LDAP is an application protocol for querying and modifying directory services
  - Runs over TCP/IP
- **Directory**
  - A set of objects with similar attributes
    - Organized in a logical and hierarchical manner
  - Example:
    - Telephone directory
      - Series of names (either of persons or organizations)
      - Organized alphabetically
      - Each name has an address and phone number
  - LDAP is often used by other services for authentication
  - **LDAP directory tree is drawn.**

LDAP – Attribute Naming

Attribute	Abbr.	Value
Country	C	NL
Locality	L	Amsterdam
Organization	O	Vrije Universiteit
OrganizationalUnit	OU	Comp. Sc.
CommonName	CN	Main server
Mail_Servers	—	137.37.20.3, 130.37.24.6, 137.37.20.10
FTP_Server	—	130.37.20.20
WWW_Server	—	130.37.20.20

Figure 5-22. A simple example of an LDAP directory entry using LDAP naming conventions.

Origin and influences

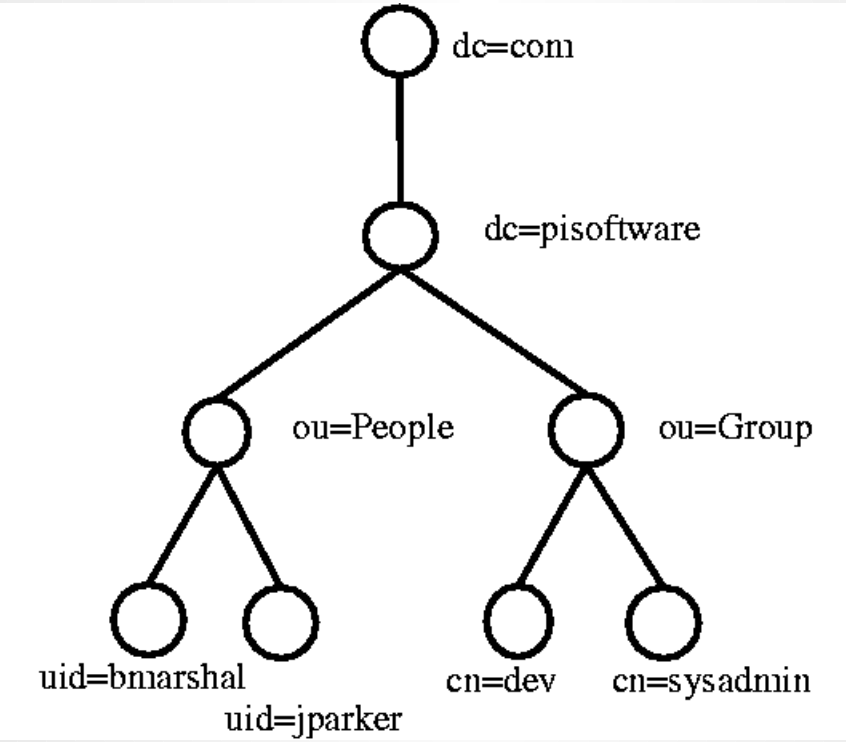
- X.500 directory services
  - Traditionally accessed via the X.500 Directory Access Protocol (DAP)
  - Required the Open Systems Interconnection (OSI) protocol stack
- LDAP originally intended to be a "lightweight" alternative protocol for accessing X.500 directory services
  - Through the simpler (and now widespread) TCP/IP protocol stack

Directory structure

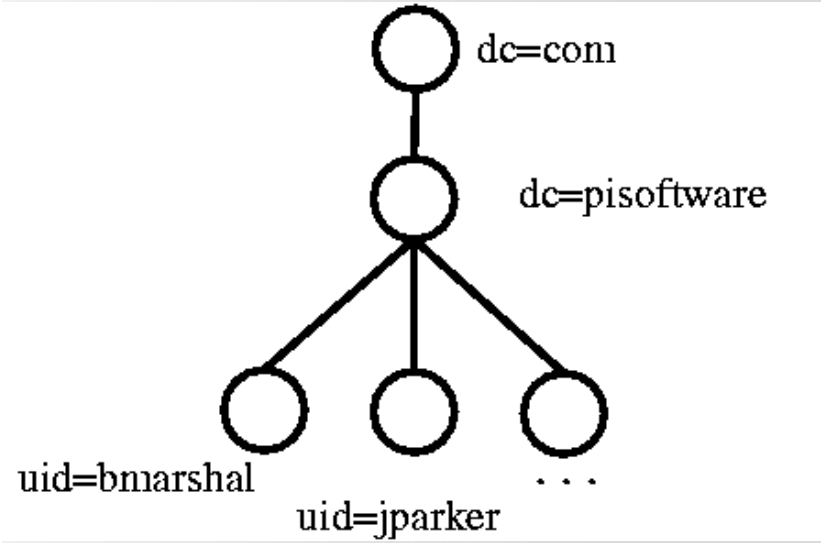
- Protocol accesses LDAP directories
  - Follows the 1993 edition of the X.500 model:
    - directory is a tree of directory entries
    - Entry consists of a set of attributes
    - An attribute has
      - a name
      - an *attribute type* or *attribute description*
      - one or more values
    - Attributes are defined in a *schema*
    - Each entry has a unique identifier:
      - *Distinguished Name* (DN)
        - Consists of its *Relative Distinguished Name* (RDN) constructed from some attribute(s) in the entry
        - Followed by the parent entry's DN
    - Think of the DN as a full filename and the RDN as a relative filename in a folder

LDAP Data Structure

Hierarchical



Flat



dc: domain component
ou: organizational unit

Protocol overview

- Client starts an LDAP session by connecting to an LDAP server
  - Default on TCP port 389
- Client sends operation requests to the server
  - Server sends responses in turn
- With some exceptions the client need not wait for a response before sending the next request
  - Server may send the responses in any order

Protocol overview

- The client may request the following operations:
  - Start TLS
    - Optionally protect the connection with Transport Layer Security (TLS), to have a more secure connection
  - Bind - authenticate and specify LDAP protocol version
  - Search - search for and/or retrieve directory entries
  - Compare - test if a named entry contains a given attribute value
  - Add a new entry
  - Delete an entry
  - Modify an entry
  - Modify Distinguished Name (DN) - move or rename an entry
  - Abandon - abort a previous request
  - Extended Operation - generic operation used to define other operations

LDAP BACKENDS

- THE BASIC DAEMON PROCESS THAT RUNS ON THE LDAP SERVER CALLED SLAPD COMES WITH THREE DIFFERENT BACKEND DATABASES
- WE ASSUME THAT IN OUR CASE WE USE LDBM THE MOST USED ONE

LDIF

- LDIF STANDS FOR LDAP DATA INTERCHANGE FORMAT
- DIRECTORY ENTRIES IN LDAP ARE IN THE FORM OF LDIF

LDIF FORMAT

- BASIC FORM OF LDIF :
  #COMMENT
  DN: <DISTINGUSHED NAME>
  <ATTRDESC>: <ATTRVALUE>
  <ATTRDESC>: <ATTRVALUE>
  .....
- EXAMPLE :
  DN: UID=ALAKESH DC=IIT DC=EDU

Reference

1. **Authors :**Wien-Verteilte_Systeme_VO_(Göschka)_-Tannenbaum

   Book name: Distributed systems principles and paradigms 2nd_edition

   Page (237/705)