**ASSIGNMENT 3**

1.Explain the different file caching schemes in DFS.

Ans:

1. A file system is a subsystem of the operating system that performs file management activities such as organization, storing, retrieval, naming, sharing, and protection of files.
2. A file system frees the programmer from concerns about the details of space allocation and layout of the secondary storage device.
3. A distributed file system provides the following types of services:
4. Storage service

Allocation and management of space on a secondary storage device thus providing a logical view of the storage system.

1. True file service

Includes file-sharing semantics, file-caching mechanism, file replication mechanism, concurrency control, multiple copy update protocol etc.

4. File-caching scheme for distributed systems:

1) Cache location

2) Modification Propagation

3) Cache Validation

1. Cache Location:This refers to the place where the cached data is stored. Assuming that the original location of a file is on its servers disk, there are three possible cache locations in a distributed file system:
2. Server’s main memory:In this case a cache hit costs one network access.

It does not contribute to scalability and reliability of the distributed file system.

Since we every cache hit requires accessing the server.

Advantages:

1. Easy to implement
2. Totally transparent to clients
3. Easy to keep the original file and the cached data consistent.
4. Client’s disk:In this case a cache hit costs one disk access. This is somewhat slower than having the cache in server’s main memory. Having the cache in server’s main memory is also simpler.

Advantages:

1. Provides reliability against crashes since modification to cached data is lost in a crash if the cache is kept in main memory.
2. Large storage capacity.
3. Contributes to scalability and reliability because on a cache hit the access request can be serviced locally without the need to contact the server.

1. Client’s main memory:Eliminates both network access cost and disk access cost. This technique is not preferred to a client�s disk cache when large cache size and increased reliability of cached data are desired.

Advantages:

1. Maximum performance gain.
2. Permits workstations to be diskless.
3. Contributes to reliability and scalability.

2) Modification Propagation:When the cache is located on clients’ nodes, a file’s data may simultaneously be cached on multiple nodes. It is possible for caches to become *inconsistent* when the file data is changed by one of the clients and the corresponding data cached at other nodes are not changed or discarded.

-There are two design issues involved:

1.When to propagate modifications made to a cached data to the corresponding file server.

2. How to verify the validity of cached data.

The modification propagation scheme used has a critical affect on the system’s performance and reliability. Techniques used include:

1. Write-through scheme.

When a cache entry is modified, the new value is immediately sent to the server for updating the master copy of the file.

Advantage:

High degree of reliability and suitability for UNIX-like semantics:This is due to the fact that the risk of updated data getting lost in the event of a client crash is very low since every modification is immediately propagated to the server having the master copy.

Disadvantage:This scheme is only suitable where the ratio of read-to-write accesses is fairly large. It does not reduce network traffic for writes.This is due to the fact that every write access has to wait until the data is written to the master copy of the server. Hence the advantages of data caching are only read accesses because the server is involved for all write accesses.

1. Delayed-write scheme: To reduce network traffic for writes the delayed-write scheme is used. In this case, the new data value is only written to the cache and all updated cache entries are sent to the server at a later time.

Advantages of delayed-write scheme:

1. Write accesses complete more quickly because the new value is written only client cache. This results in a performance gain.

2. Modified data may be deleted before it is time to send to send them to the server (e.g. temporary data). Since modifications need not be propagated to the server this results in a major performance gain.

3. Gathering of all file updates and sending them together to the server is more efficient than sending each update separately.

Disadvantage of delayed-write scheme:

Reliability can be a problem since modifications not yet sent to the server from a client’s cache will be lost if the client crashes.

3) Cache Validation schemes:1.The modification propagation policy only specifies when the master copy of a file on the server node is updated upon modification of a cache entry. It does not tell anything about when the file data residing in the cache of other nodes is updated.

2. A file data may simultaneously reside in the cache of multiple nodes. A client’s cache entry becomes stale as soon as some other client modifies the data corresponding to the cache entry in the master copy of the file on the server.

3. It becomes necessary to verify if the data cached at a client node is consistent with the master copy. If not, the cached data must be invalidated and the updated version of the data must be fetched again from the server.

4.There are two approaches to verify the validity of cached data: the client-initiated approach and the server-initiated approach.

1. Client-initiated approach

The client contacts the server and checks whether its locally cached data is consistent with the master copy.

1. Server-initiated approach
2. A client informs the file server when opening a file, indicating whether a file is being opened for reading, writing, or both. The file server keeps a record of which client has which file open and in what mode.
3. So server monitors file usage modes being used by different clients and reacts whenever it detects a potential for inconsistency. E.g. if a file is open for reading, other clients may be allowed to open it for reading, but opening it for writing cannot be allowed. So also, a new client cannot open a file in any mode if the file is open for writing.

1. When a client closes a file, it sends intimation to the server along with any modifications made to the file. Then the server updates its record of which client has which file open in which mode.
2. When a new client makes a request to open an already open file and if the server finds that the new open mode conflicts with the already open mode, the server can deny the request, queue the request, or disable caching by asking all clients having the file open to remove that file from their caches.

2. Case study: Design of Google as a distributed service.

Ans:

Introduction:

Google is a US-based corporation with its headquarter in Mountain View, CA. offering Internet search and broader web applications and earning revenue largely from advertising associated with such services.The name is a play on the word googol, the number 10^100 ( or 1 followed by a hundred zeros), emphasizing the sheer scale of information in Internet today. Google was born out of a research project at Standford with the company launched in 1998.

Google Distributed System: Design Strategy :

Google has diversified and as well as providing a search engine is now a major player in cloud computing. • 88 billion queries a month by the end of 2010. The user can expect query result in 0.2 seconds. • Good performance in terms of scalability, reliability, performance and openness.

Google Search Engine :Consist of a set of services •

1. Crawling: to locate and retrieve the contents of the web and pass the content onto the indexing subsystem. Performed by a software called Googlebot.
2. Indexing: produce an index for the contents of the web that is similar to an index at the back of a book, but on a much larger scale. Indexing produces what is known as an inverted index mapping words appearing in web pages and other textual web resources onto the position where they occur in documents. In addition, index of links is also maintained to keep track of links to a given site.
3. Ranking: Relevance of the retrieved links. Ranking algorithm is called PageRank inspired by citation number for academic papers. A page will be viewed as important if it is linked to by a large number of other pages.

Google as a cloud provider:

Google is now a major player in cloud computing which is defined as “a set of Internet-based application, storage and computing services sufficient to support most user's needs, thus enabling them to largely or totally dispense with local data storage and application software.

1. Software as a service: offering application-level software over the Internet as web application. A prime example is a set of web-based applications including Gmail, Google Docs, Google Talk and Google Calendar. Aims to replace traditional office suites. ( more examples in the following table)
2. Platform as a service: concerned with offering distributed system APIs and services across the Internet, with these APIs used to support the development and hosting of web applications. With the launch of Google App Engine, Google went beyond software as a service and now offers it distributed system infrastructure as a cloud service. Other organizations to run their own web applications on the Google platform.

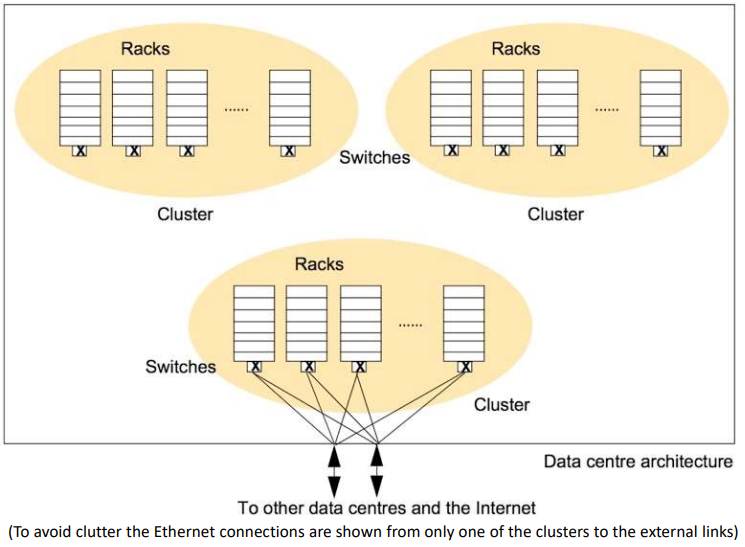
Google Physical model:

1. The key philosophy of Google in terms of physical infrastructure is to use very large numbers of commodity PCs to produce a cost-effective environment for distributed storage and computation.
2. Purchasing decision are based on obtained the best performance per dollar rather than absolute performance. When Brin and Page built the first Google search engine from spare hardware scavenged from around the lab at Standford university.
3. Typical spend is $1k per PC unit with 2 Terabytes of disk storage and 16 gigabytes of memory and run a cut-down version of Linux kernel.

Physical Architecture of Google is constructed as:

1. Commodity PCs are organized in racks with between 40 to 80 PCs in a given rack. Each rack has a Ethernet Switch.
2. 30 or more Racks are organized into a cluster, which are a key unit of management for placement and replication of services. Each cluster has two switched connected the outside world or other data centers.
3. Clusters are housed in data centers that spread around the world.

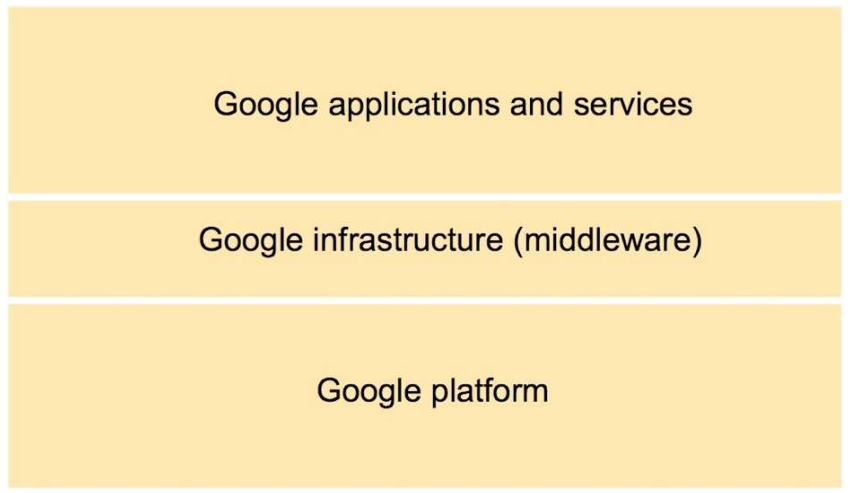
Organization of the Google physical infrastructure:



Key Requirements

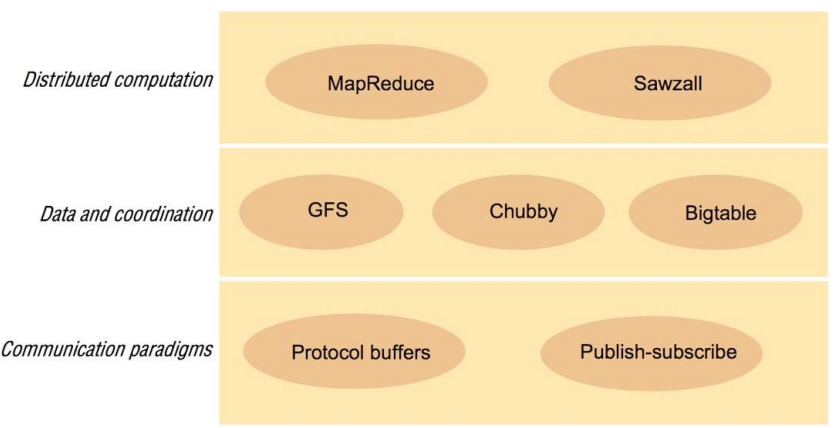
1. Scalability: i). Deal with more data ii) deal with more queries and iii) seeking better results • Reliability: There is a need to provide 24/7 availability. Google offers 99.9% service level agreement to paying customers of Google Apps covering Gmail, Google Calendar, Google Docs, Google sites and Google Talk. The wellreported outage of Gmail on Sept. 1st 2009 (100 minutes due to cascading problem of overloading servers) acts as reminder of challenges.
2. Performance: Low latency of user interaction. Achieving the throughput to respond to all incoming requests while dealing with very large datasets over network.
3. Openness: Core services and applications should be open to allow innovation and new application.

The overall Google systems architecture:



Google Infrastructure:

1. The underlying communication paradigms, including services for both remote invocation and indirect communication.
2. The protocol buffers offers a common serialization format including the serialization of requests and replies in remote invocation.
3. The publish-subscribe supports the efficient dissemination of events to large numbers of subscribers.
4. Data and coordination services providing unstructured and semi-structured abstractions for the storage of data coupled with services to support access to the data. • GFS offers a distributed file system optimized for Google application and services like large file storage.
5. Chubby supports coordination services and the ability to store small volumes of data
6. BigTable provides a distributed database offering access to semi-structure data.
7. Distributed computation services providing means for carrying out parallel and distributed computation over the physical infrastructure.
8. MapReduce supports distributed computation over potentially very large datasets for example stored in Bigtable.
9. Sawzall provides a higher-level language for the execution of such distributed computation.



Distributed Computation Services:

It is important to support high performance distributed computation over the large datasets stored in GFS and Bigtable. The Google infrastructure supports distributed computation through MapReduce service and also the higher level Sawzall language.

Carry out distributed computation by breaking up the data into smaller fragments and carrying out analyses (sorting, searching and constructing inverted indexes) of such fragments in parallel, making use of the physical architecture.

MapReduce {Dean and Ghemawat 2008} is a simple programming model to support the development of such application, hiding underlying detail from the programmer including details related to the parallelization of the computation, monitoring and recovery from failure, data management and load balancing onto the underlying physical infrastructure.

• Key principle behind MapReduce is that many parallel computations share the same overall pattern– that is:

• Break the input data into a number of chunks

• Carry out initial processing on these chunks of data to produce intermediary results ( map function).

• Combine the intermediary results to produce the final output.(reduce function)

Summary:

Infrastructure can be a competitive advantage- It certainly is for Google. They can roll out new internet services faster, cheaper, and at scale at few others can compete with. Many companies take a completely different approach. Many companies treat infrastructure as an expense. Each group will use completely different technologies and there will be little planning and commonality of how to build systems. Google thinks of themselves as a systems engineering company, which is a very refreshing way to look at building software.

3. Case Study: Network File System (NFS) features.

Ans:

Introduction:

A file system is the way in which files are named and where they are placed logically for storage and retrieval. The different operating systems have file systems in which files are placed somewhere in a hierarchical (tree) structure. Network file system is any file system that allows access to files from multiple hosts sharing via a computer network. This makes it possible for multiple users on multiple machines to share files and storage resources. NFS provides transparent and remote access to file systems. It is propitious to go for NFS as it can be machine and operating system independent, provides transparent access, provides high crash recovery mechanisms and it is highly Scalable.

There are two network file systems in widespread use today:NFS**,** originally developed by Sun, which is the dominant system in Unix and Linux systems, and CIFS or SMB for Windows systems.

Distributed File system requirements

Related requirements in distributed file systems are:

1. Transparency
2. Concurrency
3. Replication
4. Heterogeneity
5. Fault tolerance
6. Consistency
7. Security
8. Efficiency

File Service Architecture

An architecture that offers a clear separation of the main concerns in providing access to files is obtained by structuring the file service as three components:

1. A flat file service
2. A directory service
3. A client module.

The Client module implements exported interfaces by flat file and directory services on server side.

Responsibilities of various modules can be defined as follows:

1. Flat file service: Concerned with the implementation of operations on the contents of file. Unique File Identifiers (UFIDs) are used to refer to files in all requests for flat file service operations. UFIDs are long sequences of bits chosen so that each file has a unique among all of the files in a distributed system. DISTRIBUTED FILE SYSTEM File Service Architecture
2. Directory service: Provides mapping between text names for the files and their UFIDs. Clients may obtain the UFID of a file by quoting its text name to directory service. Directory service supports functions needed generate directories, to add new files to directories.
3. Client module: It runs on each computer and provides integrated service (flat file and directory) as a single API to application programs. For example, in UNIX hosts, a client module emulates the full set of Unix file operations. It holds information about the network locations of flat-file and directory server processes; and achieve better performance through implementation of a cache of recently used file blocks at the client.

# Features of the NFS Service

# This section covers the important features that are included in the NFS service.

# NFS Version 2 Protocol:

# Version 2 was the first version of the NFS protocol in wide use. Version 2 continues to be available on a large variety of platforms. All Solaris releases support version 2 of the NFS protocol, but Solaris releases prior to Solaris 2.5 support version 2 only.

## NFS Version 3 Protocol:

## An implementation of NFS version 3 protocol was a new feature of the Solaris 2.5 release. Several changes have been made to improve interoperability and performance. For optimal use, the version 3 protocol must be running on both the NFS servers and clients.This version enables safe asynchronous writes on the server, which improve performance by allowing the server to cache client write requests in memory. The client does not need to wait for the server to commit the changes to disk, so the response time is faster. Also, the server can batch the requests, which improves the response time on the server.

## The NFS version 3 protocol removes the 8-Kbyte transfer size limit. Clients and servers negotiate whatever transfer size they support, rather than conform to the 8-Kbyte limit imposed in version 2. The Solaris 2.5 implementation defaults to a 32-Kbyte transfer size.

## NFS ACL Support:

## Access control list (ACL) support was added in the Solaris 2.5 release. ACLs provide a finer-grained mechanism to set file access permissions than is available through standard UNIX file permissions. NFS ACL support provides a method of changing and viewing ACL entries from a Solaris NFS client to a Solaris NFS server. See ["Using Access Control Lists (ACLs)" in System Administration Guide: Security Services](https://docs.oracle.com/docs/cd/E19683-01/806-4078/6jd6cjs3d/index.html) for more information about ACLs.

## NFS Over TCP

## The default transport protocol for the NFS protocol was changed to the Transport Control Protocol (TCP) in the Solaris 2.5 release. TCP helps performance on slow networks and wide area networks. TCP also provides congestion control and error recovery. NFS over TCP works with version 2 and version 3. Prior to 2.5, the default NFS protocol was User Datagram Protocol (UDP).

## Network Lock Manager and NFS

## The Solaris 2.5 release also included an improved version of the network lock manager, which provided UNIX record locking and PC file sharing for NFS files. The locking mechanism is now more reliable for NFS files, so commands which use locking are less likely to hang.

## NFS Large File Support

## The Solaris 2.6 implementation of the NFS version 3 protocol was changed to correctly manipulate files larger than 2 Gbytes. The NFS version 2 protocol and the Solaris 2.5 implementation of the version 3 protocol cannot handle files larger than 2 Gbytes.

## NFS Client Failover

## Dynamic failover of read-only file systems was added in the Solaris 2.6 release. Failover provides a high level of availability for read-only resources that are already replicated, such as man pages, other documentation, and shared binaries. Failover can occur anytime after the file system is mounted. Manual mounts can now list multiple replicas, much like the automounter in previous releases. The automounter has not changed, except that failover need not wait until the file system is remounted.

## Kerberos Support for the NFS Service

## Support for Kerberos V4 clients was included in the Solaris 2.0 release. In the 2.6 release, the mount and share commands were altered to support NFS version 3 mounts that use Kerberos V5 authentication. Also, the share command was changed to enable multiple authentication flavors for different clients.

## WebNFS Support

## The Solaris 2.6 release also included the ability to make a file system on the Internet accessible through firewalls, using an extension to the NFS protocol. One of the advantages to using the WebNFSTM protocol for Internet access is its reliability. The service is built as an extension of the NFS version 3 and version 2 protocol. Also, an NFS server provides greater throughput under a heavy load than Hypertext Transfer protocol (HTTP) access to a Web server. This throughput can decrease the amount of time that is required to retrieve a file. In addition, the WebNFS implementation provides the ability to share these files without the administrative overhead of an anonymous ftp site.

## RPCSEC\_GSS Security Flavor

## A security flavor, called RPCSEC\_GSS, is supported in the Solaris 7 release. This flavor uses the standard GSS-API interfaces to provide authentication, integrity, and privacy, as well as enabling support of multiple security mechanisms..

## Solaris 7 Extensions for NFS Mounting

## The Solaris 7 release includes extensions to the mount command and automountd command that enable the mount request to use the public file handle instead of the MOUNT protocol. The MOUNT protocol is the same access method that the WebNFS service uses. By circumventing the MOUNT protocol, the mount can occur through a firewall. In addition, because fewer transactions need to occur between the server and client, the mount should occur faster.

## The extensions also enable NFS URLs to be used instead of the standard path name. Also, you can use the public option with the mount command and the automounter maps to force the use of the public file handle.

## Security Negotiation for the WebNFS Service

## A new protocol has been added to enable a WebNFS client to negotiate a security mechanism with an NFS server in the Solaris 8 release. This protocol provides the ability to use secure transactions when using the WebNFS service.

## NFS Server Logging

## In the Solaris 8 release, NFS server logging enables an NFS server to provide a record of file operations that have been performed on its file systems. The record includes information to track what is accessed, when it is accessed, and who accessed it. You can specify the location of the logs that contain this information through a set of configuration options. You can also use these options to select the operations that should be logged. This feature is particularly useful for sites that make anonymous FTP archives available to NFS and WebNFS clients.

## Autofs Features

## Autofs works with file systems that are specified in the local name space. This information can be maintained in NIS, NIS+, or local files.

## A fully multithreaded version of automountd was included in the Solaris 2.6 release. This enhancement makes autofs more reliable and enables concurrent servicing of multiple mounts, which prevents the service from hanging if a server is unavailable.

## The autofs service supports browsability of indirect maps. This support enables a user to see what directories could be mounted, without having to actually mount each one of the file systems. A -nobrowse option has been added to the autofs maps, so that large file systems, such as /net and /home, are not automatically browsable. Also, you can turn off autofs browsability on each client by using the -n option with automount.

Summary

Using the NFS protocol, you can transfer files between computers running Windows and other non-Windows operating systems, such as Linux or UNIX.NFS in Windows Server includes Server for NFS and Client for NFS. A computer running Windows Server can use Server for NFS to act as a NFS file server for other non-Windows client computers. Client for NFS allows a Windows-based computer running Windows Server to access files stored on a non-Windows NFS server.