# Lecture 23: April 23
# CMPSCI 677 Operating Systems - Spring 2018
# Lecturer: Prashant Shenoy
# Author: Krishna Prasad Sankaranarayanan

April 28, 2018

## 1 XFS

XFS combines two main concepts ; RAID - Redundant Array of Inexpensive Disks) and Log Structured File Systems (LFS). It uses a concept of Network Stripping and RAID over a network wherein, a file is partitioned into blocks and provided to different servers. These blocks are then made as a Software RAID file by computing a parity for each block which resides on a different machine.

In log structured File systems, data is sequentially written in the form of a log. The motivation for LFS would be the large memory caches used by the OS. Larger, the size of cache, more the number of cache hits due to reads, better will be the payoff due to the cache. The disk would be accessed only if there is a cache miss. Due to the this locality of access, mostly write requests would trickle to the disk. Hence, the disk traffic comes predominantly from write. In traditional hard drive disks, a disk head read or writes data . Hence, to read a block, a seeks needs to be done ie move the head to the right track on the disk.

How to optimize a file system which sees mostly write traffic ?

The basic insight is to reduce the time spent on seek and waiting for the required block to spin by. Every read/write request incurs a seek time and a rotational latency overhead. In general , random access layout is assumed for all blocks in the disk wherein the next block is present in an arbitrary location. This would require a seek time.

To eliminate this, a sequential form of writing facilitated by LFS can be used. The main idea of LFS is that we try to write all the blocks sequentially one after the other. Thus LFS essentially buffers the writes and writes them in contiguous blocks into segments in a log like fashion. This will dramatically improve the performance. Any new modification would be appended at the end of the current log and hence, overwriting is not allowed. Any LFS requires a garbage collection mechanism to de-fragment and clean holes in the log.

Hence, XFS ensures 1. fault tolerance - due to RAID, 2. Parallelism - due to blocks being sent to multiple nodes. 3. High Performance - due to Log structured organization.

In SSD's, the above mentioned optimization to log structures doesn't give any benefits since there are no moving parts and hence, no seek.

### 1.1 Combine LFS with Software RAID

Log written sequentially are chopped into blocks which a parity groups. Each parity group becomes a server on a different machine in a RAID fashion

# 2 HDFS - Hadoop Distributed File system

It is designed for high throughput - very large datasets. It optimizes the data for batch processing rather than interactive processing. HDFS has a simple coherency model in which it assumes a WORM (Write Once Read Many) model. In WORM, file do not change and changes are append-only.

## 2.1 Architecture

There are 2 kinds of nodes in HDFS ; Data and Meta-data nodes. Data nodes store the data whereas, meta-data keeps track of where the data is stored. Average block size in a file system is 4 KB. In HDFS, due to large datasets, block size is 64 MB.Replication of data prevents disk failures. Default replication factor in HDFS is 3.

# 3 GFS - Google File System

Master node acts as a meta-data server. It uses a file system tree to locate the chunks (GFS terminology for blocks). Each chunk is replicated on 3 nodes. Each chunk is stored as a file in Linux . file system.

# 4 Distributed Middleware

## 4.1 Distributed Objects

In case of remote objects, clients have a stub called proxy. Objects that needs to be accessed is on a different server. Hence, an in an RMI call, client code is calling a method on a object on a different machine. RPC's inherently are not object oriented. RMI's make RPC's object oriented. RPC's do not allow network pointer passing.

On the other hand, distributed objects are itself partitioned or replicated across different machines. Distributed objects use RPC as well as allow object reference lookups ie network pointers unlike in RPC's.

Middleware systems were developed originally for complex distributed applications but nowadays, they are utilized in a client-server framework. Since they were developed for complex applications,making them heavy weight, some became commercial failures : eg. CORBA.

## 4.2 Example - Object References

This section shows a CORBA object reference. It mainly contains the Host, Socket port and Object key used for memory referencing.

## 4.3 EJB - Enterprise Java Beans

It is basically standard Java added with services like RMI, JDBC - used to connect to Databases, JMS - Java Messaging Service etc.

Middleware services also provide entity beans which are essentially persistent objects. The lifetime of a persistent object is independent of the lifetime of the process. The state of the object is stored on the disk which can be reconstructed when necessary.

### 4.3.1 Types of EJB's

1. Stateless Session beans - Object stays alive for the duration of the session. It has only methods.
2. Stateful Session Beans - Closest to Java objects with state and private variables.
3. Entity beans - persistent objects.
4. Message -driven beans - used to send/receive messages between one another.

## 4.4 CORBA - Common Object Request Broker Architecture

Object Request broker (ORB) is a intermediate communication channel that allows communication between objects. The advantage is that the number of services provided by CORBA middleware is extensive. On the other hand, the overhead becomes high and is apt only for complex applications.

### 4.4.1 Object Model

Interface Definition Language (IDL) Proxy is used to specify the objects and services. Object adapter provides portability between languages. Thus CORBA, is language independent.

### 4.4.2 Object Invocation Models

Failure Semantics

1. Synchronous - at most once
2. One-way - best effort
3. Deferred Synchronous - at most once

### 4.4.3 Event and Notification Service

This services is used to deploy PUB-SUB applications. Event channel connects Publishers to Subscribers in CORBA. It is a push -push model where data is pushed from Supplier to Event channel and subsequently to Consumer.

In pull-pull mode, event channel polls data from the Supplier and similarly, Consumer pulls data from the event channel. To reduce polling, Asynchronous method invocation is used wherein a notification after data is arrived would be given. With respect to pure message overhead : Pure Push $<$ Asynchronous Pull $<$ Pure Pull.

## 4.5 DCOM - Distributed Component Object Model

It is Microsoft's middleware which has now evolved into .NET. .Net has a language independent runtime.

COM is a simple RMI based framework running local to a machine. It is mainly used for communication between Microsoft applications via embedding and document linking. These applications communicated via COM RPC Framework. This feature is called Object Linking and Embedding (OLE).The ActiveX layer facilitates exposing these services as web applications.

### 4.5.1 Type Library and Registry

Objects can be persistent on disks. Persistent objects are called Monikers.

## 4.6 Distributed Coordination System / Blackboard Architecture

Distributed applications can be classified based on whats happening in time and space dimension. Applications can either be coupled or decoupled in space and time.

1. Direct
2. Mailbox - receiver is known but receiver state does not matter.
3. Meeting Oriented
4. Generative Communication - components can communicate with another without knowing who might read it or when it would be read. Hence, loosest form of communication.

### 4.6.1 Jini

It facilitates service discovery. These are also 0-Configuration Services. eg: Wifi. It uses a blackboard or bulletin architecture. Services advertise on the bulletin board and machines can access the services it requires through the board. In Jini, Bulletin board is called JavaSpace or Tuple Space. Each tuple is a Java object. Essentially, Jini utilizes a Pub-Sub archutecture with both Pull based as well as notification based discovery.

JavaSpaces can either be fully replicated or distributed. In case of replicated JavaSpaces , Writes need to be broadcasted to all replicas whereas reads are local. On the other hand, in distributed bulletin board, each board has a subset of nodes. Hence, Writes are local and reads need to be done on each and every board.