READ COPY UPDATE: TO SOLVE CONCURRENCY PROBLEM USING THREADS

Swe Swe Aung, Swe Swe Shein
Computer University (Taunggyi), Myanmar
sashiway@gmail.com, ssksucsy@gmail.com

ABSTRACT

As concurrency systems expand and become more and more popular, there is a growing need for efficient, tolerating stale data and reducing in concurrent writing problem. The use of multiple threads is beneficial in concurrent access files on file server. Threads allow operations from multiple clients to run concurrently and possibly access the same objects. The focus of this thesis is to implement concurrency. The client's requests are processed by Read_Copy Update techniques and Round Robin fashion and multiple worker threads that operate reading and writing operations on data files in file server. This paper analyses performance evaluation on processing time by single worker thread and multiple worker threads and the results show less processing time in required for multiple worker threads.

Keywords: Read_Copy Update, Multithreading, File Server.

1. INTRODUCTION

Concurrency is a property of system in which several computational processes are executing at the same time, and potentially interacting with each other. The concurrent processes may be executing truly simultaneously, in the case that they run on separate processors, or their execution steps may be interleaved to produce the appearance of concurrency [6].

Distributed file system supports the sharing of information in the form of file throughout on internet. File systems are designed to store and manage large number of files, with facilities for creating, updating, naming and deleting files. The file systems also take responsibility for the control of access to files, restricting access to files according to user's authorizations and the type of access requested(reading, updating, executing and so on).

A well-designed file service provides access to files stored at a server with performance and reliability similar to files stored on local disks. A distributed file system enables programs to store and access remote file exactly as the do local ones, allowing users to access files from any computer in an internet [3].

2. MOTIVATION

If a set of processes is accessing files with locking, then Read_Copy Update is not necessary. In the situation where the user wishes to update the file being accessed by other without lock, then Read_Copy Update is needed to access it concurrently.

This system proposes Read_Copy Update technique to solve concurrent writing problem. And Round_Robin scheduling algorithm is applied to reduce starvation and nonpreemptive.

This paper intends to implement concurrency by using Read_Copy Update technique and threads that can perform efficiently and effectively in tolerating stale data and reducing process's waiting time.

3. RELATED WORK

Operating System such as Linux often performs expensive synchronizations in common code to protect against infrequent destructive modifications. These synchronization operations on the common code paths result in increased overhead, reduced scalability on a multiprocessor [5].

This paper uses Read_Copy Update technique to determine when update operations may be safely carried out in order [4].

Copying of the shared data is permitted so that waiting time can be reduced and even eliminated. The copies of the shared data will reside in a set of buffers. The writer, when it wishes to change the shared data, will write into a subset of the buffers. A reader will obtain a correct, recent value from one of the buffer [2].

4. THEORETICAL BACKGROUND

4.1 File Server

A computer connected to the network that contains primary files/applications and shares them as requested with the other computers on the network. If the file server is dedicated for that purpose only, it is connected to a client/server network, e.g. Novell Netware. All the computers connected to a peer-to-peer network are capable of being the file server, e.g. LANtastic and Windows for Workgroups [7].

4.2 Concurrency in File Server

A thread package with multiple independent threads of execution within a single process supports multiple clients and periodic operations within file server.

File Server creates a thread for a client when it requests to access file. Thread is deleted when client terminates its connection.

Since server process may be processing the requests of hundreds of clients simultaneously, the server operates in real-time [3].

4.3 File Server Architecture

In the client/server model, a file server is a computer responsible for the central storage and management of data files so that other computers on the same network can access the files.

A file server allows users to share information over a network without having to physically transfer files by floppy diskette or some other external storage device. Any computer can be configured to be a host and act as a file server.

In its simplest form, a file server may be an ordinary PC that handles requests for files and sends them over the network. In a more sophisticated network, a file server might be a dedicated network-attached storage (NAS) device that also serves as a remote hard disk drive for other computers, allowing anyone on the network to store files on it as if to their own hard drive [8].

4.4 Read_Copy Update

It is updating a copy of an element while allowing concurrent reads on it. Any reading thread that starts its access after an update completes is guaranteed to see the new data.

Old data may be flagged so that the reading threads may detect it and take explicit steps to obtain up-to-date, if required [4].

4.5 What is a Thread?

A thread can be loosely defined as a separate stream of execution that takes place simultaneously with and independently of everything else that might be happening.

A thread is like a classic program that starts at point A and executes until it reaches point B. It does not have an event loop. A thread runs independently of anything else happening in the computer.

Without threads an entire program can be held up by one CPU intensive task or one infinite loop, intentional or otherwise. With threads the other tasks that don't get stuck in the loop can continue processing without waiting for the stuck task to finish [8].

4.6 Multithreading

Multithreading as a widespread programming and execution model allows multiple threads to exist within the context of a single process.

These threads share the process' resources but are able to execute independently. The threaded programming model provides developers with a useful abstraction of concurrent execution.

However, perhaps the most interesting application of the technology is when it is applied

to a single process to enable parallel execution on a multiprocessor system [8].

4.7 Round Robin Policy

A small unit of time, called a time quantum is defined. The ready queue is treated as a circular queue. New processes are added to the tail of the ready queue. The scheduler picks the first processes from the ready queue, sets a timer to interrupt after 1 time quantum, and dispatches the process [1].

5. OVERVIEW OF THE SYSTEM

5.1 System Architecture

This system develops a concurrent system in which processes can run in parallel. In this system file client and file server architecture are configured in network environment and overview of system architecture can be seen in Figure 1.

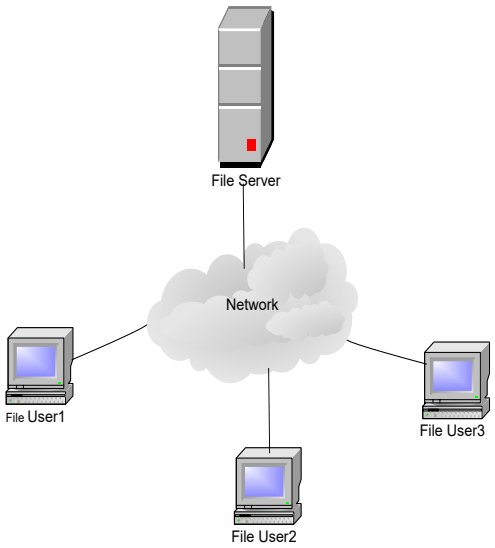

Figure 1. Overview of the System Architecture

There are five cooperated process modules to simulate this system. They are File Client, Request Pool, Worker Pool, PCB and File Server modules and these components can be shown as block diagram in Figure 2.

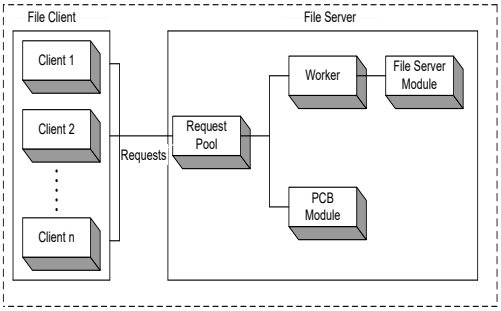

Figure 2. Block Diagram of System

The main components of this system are the cooperated modules working together at file server. The detail architecture of these components can be designed in Figure 3.

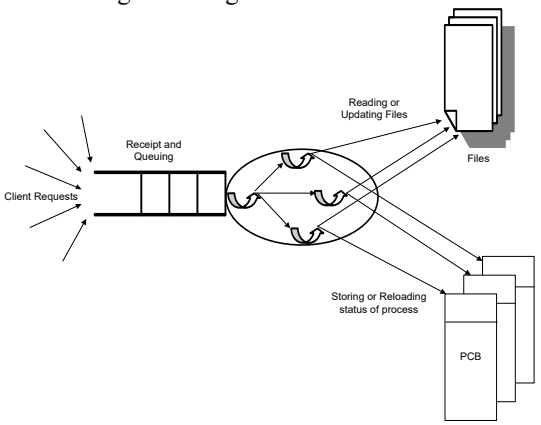

Figure 3. Overview Architecture of the File Server

In this figure, every file request from any file user, who wants to access a file stored in file server, is stored into Request Pool. On the other hand, the worker threads in Worker Pool retrieve the request and process the corresponding operations on related files. The main attributes of request can be shown as follows in Figure 4.

requestID	requestFile	accessType	data	sourceID

Figure 4. Definition of A Request

Whenever a worker thread retrieves and processes a request, the PCB module creates a process control block (PCB) for each request. The PCB structure has the following attributes as shown in Figure 5.

ThreadID
RequestID
TargetFile
AccessType
NewFile
NewFilePtr
DataFile
DataFilePtr

Figure 5. PCB architecture

The PCB module is critical in implementing the concurrent processes in multithreading architecture. Since the worker threads for each request are processed with round robin fashion. Whenever the time quantum for each thread is finished, the process's latest conditions have to be saved and need to recognize that status for resuming. So, this system prepares the storage area for conditional factors presented in Figure 5. In which the Target File means the requested file to access by file client. Access Type is what the client wants to do on file. The Access Type here may be read or write access. New File is essential for read-copy-update semantic, when in which read or write access is processed; the original file is coped to another location for concurrent accesses.

While request is processing before completion, it works with New File and update the New File to Target File after processing has finished. NewFilePtr is requested to notice the previous work and is used to resume the process. When the file access is 'write' the DataFile maintains the data to be write on the TargetTile and Data FilePtr is also needed for resume case.

The File Server module maintains the data files and provides the file services such as read and writes accesses.

The workflow of system's processes can be seen in Figure 6 and the working procedure can be illustrated by the algorithm in next section

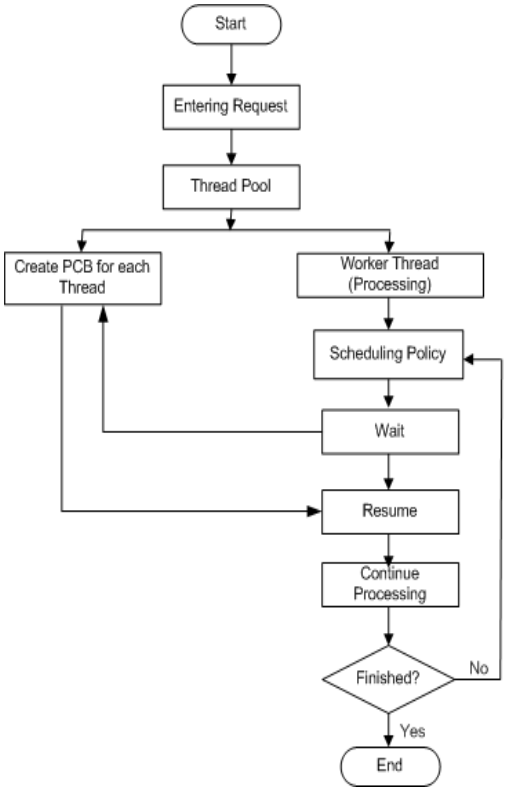

Figure 6. Process Flow Diagram

5.2 File Server Algorithm

To illustrate the flow of system's processes, this system needs to identify the following notations for system parameters.

Let *RP* be request pool which receives request from clients.

$r_i \in RP$ $(i=1,2,\ldots,n)$

Let *TP* be threads pool in which threads for requests.

Assume scheduler executes at least once every 100 milliseconds.

Scheduler handles 10 requests per second.

$t_j \in TP$ $(j=1,2,\ldots,n)$

Let *PCB* be process control block which records the latest status of each process.

$c_j \in PCB$ ($j=1,2,...,n$)
Let F be file server in which files exist.
$f_k \in F$ ($k=1,2,...,m$)

While (*RP* is true)
{ 1.create thread t_j for each request r_i
2. create c_j for each t_j
3. do concurrent process
if access type of t_i is write and new
copy f_k to f_k'
while(time is valid)
{ if access type of t_j is write
{ if state of t_j is new
{ write data to f_k'
if process complete
Delete c_j }
if state of t_j is resume
{ reload t_j with c_j attributes
continue writing to f_k' if process complete
Delete c_j }
}//Write
if access type of t_j is read
{ if state of t_j is new
{ read data from f_k
if process complete
Delete c_j
}
if state of t_j is resume
{ continue reading from f_k
if process complete
Delete c_j
}
}//Read
}//timer
4.Switch to next thread
5.Go to step 3.

6. PERFORMANCE ANALYSIS

The feature of file server is processing requests by using multiple worker threads. According to nature of thread, requests processed by threads can be performed concurrently. In order to show the strong point of concurrent theory, although system performance can be better on more concurrent work, the system can degrade the performance when concurrent threads increase because the system's computing time is spent by thread creation and deletion. In this system ten worker threads are performing the requests.

Table 1. Evaluation Data on Processes Increase

No of Process	Processing Time Assume(Millisecond)	
	Single Thread	Ten Worker Threads
10	343	109
20	593	312
30	796	390
40	993	402

This paper analyses hybrid processes of read access and write access on file size of 10 MB. The experimental results can be shown in Table 1 and the comparison graph for processing time of given number of hybrid processes on single worker thread processing and multiple worker threads processing can be depicted in Figure 7.

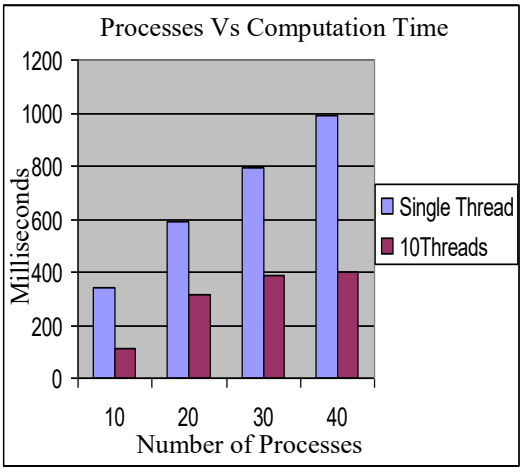

Figure 7. Performance Analysis on increasing processes

According to the experimental results, multi worker threads require less extra processing time when ten times of concurrent processes increase.

7. CONCLUSION

In this system, Read_Copy Update technique which provides read reduction in processes' waiting time and tolerates stale data when read and/or write access on file without lock concurrently. The scheduling algorithm is proposed for allocating access request to read or write text file on the basic of (RRB) policy with a small unique time quantum to be concurrent access. And in this system multithreading function is also used for creating a new thread of execution within an existing process rather than starting a new process to begin a function. We have compared the computation performance between a single thread and ten threads on the number of processes increasing. We have presented measurements that demonstrate using multithread that reduces much more processing time than using single thread. We have tested the performance of the system by using reading text file processes.

8. REFERENCES

[1] Abraham s., G. Peter Bare and G. Gagne, "Operating System Concepts", Sixth Edition ISBN-0-471-41741-2, 2002.

[2] Allan Borodin, Brett Fleisch, "Concurrent Reading While Writing", Acm Transactions on Programming Language and System, Vol.5, No.1, January 1983.

[3] George Coulouris Jean Dollimore Tim, "Distributed System Concepts and Design", Third Edition, ISBN 0-201-619/8-0, www.pearsoneduc.com

[4] P.E.Mckenney, "Read_Copy Update: Using Execution History to Solve Concurrency Problem", Senior Member,IEEE, and J.K.Slingwine.

[5] Paul E.Mckenney, "Read Copy Update", Lunux Texhnilogy Center,IBM Beaverton, http://www.rdrop.com/users/paulmck

[6] "Concurrency (Computer Science)", http://en.eikipedia.org/wiki/Concurrency_(computer_science).

[7] "File Server", http://en.eikipedia.org/wiki/File_Server

[8] "Thread (Computer Science)", http://en.eikipedia.org/wiki/Thread_(ComputerScience)