International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 05 Issue: 05 | May-2018 www.irjet.net p-ISSN: 2395-0072

Implementation of High Computational Model Using Splitting and Concatenate Approach on Raspberry Pi

Miss. Pallavi Khude[1], Tejashree Solanki[2], Pooja Panhale[3], Yuvraj Sawant[4], Gauri Mane[5]

[1,2,3,4,5] *Department of Computer Engineering, Pune/D.Y.Patil College Of Engineering Akurdi*

Abstract – *The technologies of persistent memory, like as RAM and Hard Disk, provide opportunities for preserving files in memory. Traditional file system structures may need to be re-studied. Even though there are several file systems proposed for memory, most of them have limited performance without fully utilizing the hardware at the processor side. The processes running concurrently on these processors are continuously competing for the shared resources, not only among cores, but also within the core. While resource sharing increases the resource utilization, the interference among processes accessing the shared resources can strongly affect the performance of individual processes and its predictability. In this scenario, process scheduling plays a key role to deal with performance and fairness. In this work we present a process scheduler for IOT processor that simultaneously addresses both performance and fairness. This is a major design issue since scheduling for only one of the two targets tends to damage the other. To address performance, the scheduler tackles bandwidth contention at the cache and main memory of IOT. To deal with fairness, the scheduler estimates the progress experienced by the processes, and gives priority to the processes with lower accumulated progress. Our System framework based on a new concept i.e. Data Transmission based on Space Utilization Concept" using Splitting and Concatenate Approach.*

***Key Words*: Concurrency, Multiprocessing, Scheduling, Synchronization, Main memory, File organization, Network communication, Distributed file systems .**

1. INTRODUCTION

In this work, we deal with efficient load balancing between the different resource nodes that process the client tasks, in a secure way as well as the elimination of possible single point of failure in a semi centralized load balancing architecture. To ensure that the two fundamentals i.e. co-ordination(the right things) and synchronization(the ideal time)of the processes will be executed we use synchronization algorithms. With such synchronization algorithms security will be provided to the data while transmission. This leads to less time consumption as the tasks are been executed concurrently.

Our System is a mixture of distribution model for P2P network. Data Sharing System, which has attracted the largest number of users, is the main application scheme for P2P file sharing. In broadcasting network, a single file is shared by numerous clients. The global data(files) to be transmitted is divided into Chunks(i.e. breaking the files into pieces) using chunking mechanism. The chunks can be of fixed size or variable size. All the parts connects to a central node called tracker to get a list of parts. Once all the distributed pieces are obtained at single location then whole data is successfully broadcasted to destination path.

2. System Description:

Input: Collection of Bundles of different kinds of file data.

Output: Efficient Storage utilization on remote storage.

Content verification object: Here the user's personal details like contact number and other information is verified first.

Acceptance object: The data entered by the user is being accepted by the system and stored in the database.

Termination and Clearance object: If the data is verified and accepted the registration is terminated by the submission option otherwise if the user want to edit the information he/she can select the clear option.

Unique key verification object: User need to enter username and password and then authentication of the credentials will be done whether the user is authorised user or not.

Splitting object: It converts the information into number of blocks with the help of splitter.

Concatenation object: It converts Split Blocks into combine block.

Memory object: In this, memory blocks management is processed in multi core system.

1.3 Hardware and Software Specifications

Hardware Resources Required

(a) Hardware : Dual Core or any multicore processor

(b) Speed : 1.1 Ghz minimum

(c) RAM : 1 GB or More

(d) Hard Disk : 80 GB

(e) Connector : RJ45 for LAN connectivity

(f) Key Board : Standard Windows Keyboard

(g) Cable : LAN cross connected cords

(h) Monitor : CRT/TFT

Software Resources Required

(a) Operating System : Windows

(b) UI Technology : Swing and AWT Components

(c) IDE : My Eclipse

(d) Java Version : J2SDK1.5 or later

(e) Network : Ad-HOC(Wireless) or TCP-IP(Wired)

(f) Database : MySQL

(g) Technology : Java, JZEE

(h) External Tools : CPUZ and HWMonitor

3. Implementation

Chunking Approach

Chunking refers to an methodology for making more better use of short-term memory by grouping information. Chunking breaks up long information into units or chunks. The resulting chunks are easier to commit to memory than a longer uninterrupted string of information.

Chunking is used most commonly to organize or classify large amounts of information, even when there are no obvious patterns.

Chunking can be performed using two approaches:

Fixed-Size Chunking:

Fixed-size chunking method splits files into equally sized chunks. The chunk boundaries are based on offsets like 4, 8, 16 kb, etc. [11, 12]. This method effectively solve issues of the file-level chunking method: If a huge file is altered in only a few bytes, only the changed chunks must be reindexed and moved to the backup location. However, this method creates more chunks for larger file which requires extra space to store the metadata and the time for lookup of metadata is more.

Variable Size Chunking:

The files can be broken into multiple chunks of variable sizes by breaking them up based on the content rather than on the fixed size of the files. This method resolves the fixed chunk size issue. When working on a fixed chunking algorithm, fixed boundaries are defined on the data based on chunk size which do not alter even when the data are changed. However, in the case of a variable-size algorithm different boundaries are defined, which are based on multiple parameters that can shift when the content is changed or deleted. Hence, only less-chunk boundaries need to be altered. The parameter having the highest effect on the performance is the fingerprinting algorithm.

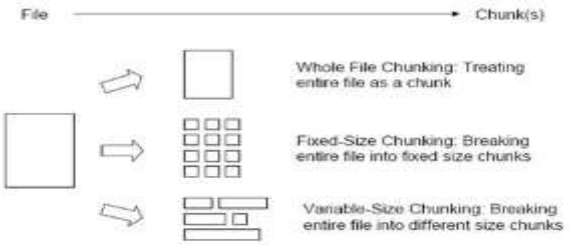

Fig 1: Chunking Methodologies

Security Mechanism:

Encoding:

For security purpose a simple approach is used that is **Bit Rotation Algorithm(BRA).**

Step1: Chunk the data using variable or fixed size chunking.

Step2: Use **BRA** that means just shuffle the bits from their original position.

Step3: Transfer the chunks

Decoding:

The exact opposite approach is performed to get the required data chunk.

Step1: Shuffle the bits in their original position.

Step2: Original Data chunks received

GUI of the System

Fig: 2 Shows the Representation of the system which copies a file from the source to destination. It also makes use of the splitting and concatenating approach to finish its task. The main aim of this was to optimize the memory Functionality of processors having small size

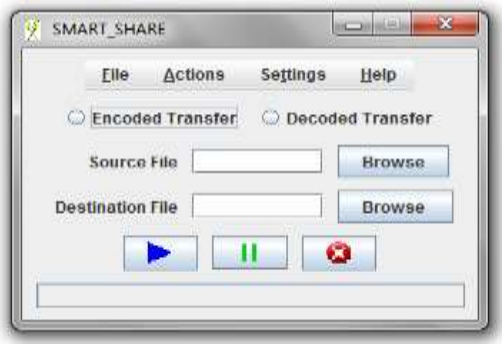

Fig 2: GUI of System

Splitting and Concatenate Approach:

Splitting approach includes chunking the file to be transferred equal to the memory available in the hard disk.

Concatenate approach includes when the hard disk memory becomes free the splitted file chunks are merged into one file.

Input:

All types of files can be taken as input for this system.

Output:

Proper memory optimization of Raspberry Pi is the goal.

4. Architecture Design:

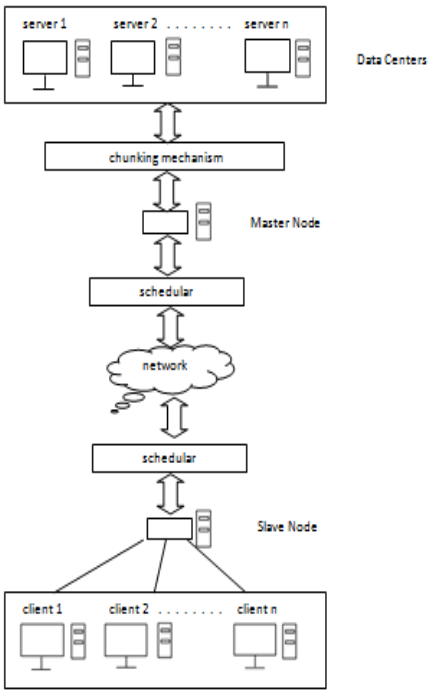

Fig3: Architecture design

5. Analysis Results of the System:

Test ID	Test case	Objectives	Steps	Input Data	Expected Result	Actual Result	Status
TC_1	Valid Password	Check Password is valid or not	Enter Password	Password	Password should be correct	Correct Password	Pass
TC_2	Login Button	Login successful	Click on Login Button	N/A	Login Successfully	Login Successfully	Pass
TC_3	Change password	Check that your password is change or not	1. Click on Change password button. 2. Type old, new password and confirm it.	Old=Password New=Password	Password should be Change.	Password is change.	Pass

Fig 4: Analysis Result 1

Test ID	Test case	Input Data	Expected Result	Actual Result	Status
TC_1	Browse Button	N/A	It should show the drives and folders in that system.	Shows the drives and the folders in the system.	Pass
TC_2	Start button for copy process	N/A	Copy process should start.	Copy process Started.	Pass
TC_3	Pause Button	N/A	Process should be Paused.	Process is paused.	Pass
TC_4	Join	N/A	Joining process should start.	Join completed	Pass.
TC_5	Encoded Transfer	Password	Encoded Transfer should start.	Encoded data transfered.	Pass
TC_6	Decoded Transfer	Password	Decoded Transfer should start.	Decoded data transfered.	Pass
TC_7	Stop Button	N/A	Process should be terminated.	Process is terminated.	Pass

Fig5: Analysis Result 2

6. CONCLUSIONS

In our paper we have considered the buffered approach to transfer the data from one destination to another. The data can be of any type be it application files, Audio , Video, Text , Images ,Pdf, etc. Since it's a buffered approach the computation time required is reduced to minimum possible milliseconds. We have used splitting and concatenation mechanism which uses chunking algorithm to efficiently solve the problem of space utilization. This obtains all memory blocks available for data sharing and hence data de-duplication is obtained. Using our new data structure, the data owner can perform insert, modify or delete operations on file blocks with high efficiency. Bit Rotation Algorithm is used for providing encryption and decryption facility while performing the transfer which is optional.

The future scope of the project can be enhanced by using our application for space utilization in cloud database management.

REFERENCES

[1] Bart Jacob, (June 2003),TSO Redbooks Project Leader, IBM,"Grid computing: What are the key components? - Taking advantage of Grid Computing for Application Enablement".

[2] Baru, C., Moore, R., Rajasekar, A. and Wan, M., (1998), "The SDSC Storage Resource Broker. 8th Annual IBM Centers for Advanced Studies".

[3] Venish and K. Siva Sankar,(2016) , "Study of Chunking Algorithm in Data Deduplication" , Proceedings of the International Conference on Soft Computing Systems, Advances in Intelligent Systems and Computing 398.

[4] Beynon, M., Kurc, T., Catalyurek, U., Chang, C., Sussman, A. and Saltz, J. (2001) , "Distributed Processing of Very Large Datasets with Data Cutter". Parallel Computing, 27 (11). 1457-1478. International Journal of Grid Computing & Applications (IJGCA) Vol.2, No.4, December 2011 61.

[5] Kurc, T., Catalyurek, U., Chang, C., Sussman, A. and Saltz, J. (2001), "Exploration and Visualization of Very Large Datasets with the Active Data Repository". IEEE Computer Graphics & Applications, 21 (4). 24-33.

[6] Chervenak, A., Foster, I., Kesselman, C., Salisbury, C. and Tuecke, S, (2001), "The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Data Sets", J. Network and Computer Applications (23). 187-200.

[7] Foster, I. and Kesselman, C. (2001), "A Data Grid Reference Architecture", Technical Report GriPhyN-2001-12.

[8] Stockinger, H., Samar, A., Allcock, B., Foster, I., Holtman, K. and Tierney, B, (2002), "File and Object Replication in Data Grids". Journal of Cluster Computing, 5 (3). 305-314.