RESEARCH ARTICLE OPEN ACCESS

A Review on High Computational IOT Device through Splitting and Concatenate Approach

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 emerging technologies of persistent memory, such as RAM and HDD , 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 parallely 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 impartialness, 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.

Keywords **— Scheduling, Splitting-Concatenate Mechanism, Performance Estimation, File System , Data Sharing, Load Balancing.**

I. INTRODUCTION

In HCI(Human-Computer Interaction)cut, paste and copy, pastes are related commands that other a user-interface interposes communication technique for transferring data. In both cases the selected data is placed in a clipboard and file operations are performed like insertion and deletion.

Our proposed system is specially designed to handle a number of deletions linear in the length of file for different operations where space utilization, security mechanism, splitting and concatenation operations are performed on file information. The proposed file system structure will fully utilize the memory in Raspberry Pi.

A new approach named "Splitting and Concatenate Approach" is introduced. This approach will help in storing a single file in multiple blocks of memory which in turn will lead to full utilization of memory. The system includes audible notification for showing the data transmission status which will allow the user to keep track of system performance even though the user is not present in front of the system. In this file system security mechanism is implemented by not allowing unauthorized user to access sensitive data. This is achieved by means of authentication and identification. The system will achieve minimum time spam and high throughput with multiple IOT system compute memory. It is to be developed by using JAVA Swing in the front end and MySQL in the back-end. Three Operating Systems will be dealt in this viz. Raspbian, Windows and Linux.

II. . BASIC CONCEPT

The purpose of this idea is to present a detailed description of the Space Utilized Based Data Sharing Scheme for IOT system using the Splitting and concatenation mechanism. It will state the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it will operate and how the system would react to external stimuli. The idea is intended for both the stakeholders and the developers of the system. This system will be designed to maximize the productivity by providing tools to assist the sender and receiver to transmit their data on small scale IOT system. By maximizing the Data Sharing work efficiency and production the system will meet the retailers needs while remaining easy to understand and use.

Related Work

User Classes and Characteristics

[1] Educational level:

Users should be comfortable with the English language.

[2] Experience:

Users should have prior information regarding the online examinations.

[3] Skills:

Users should have basic knowledge and should be comfortable using general
purpose applications on computers.

The Operating Environment
used to develop this system are
Java, awt, Swing

JAVA:

Java is a programming language expressly designed for use in the distributed environment of the Internet. Java is used to create full applications that may run on a single computer or be distributed among servers and clients in a network. It can also be used to build a small application module or applet for use as part of a Web page. Applets make it possible for a Web page user to interact with the page. It is used to build application module for use as a part of Web page and provides interaction between the pages.

AWT:

Java AWT (Abstract Window Toolkit) is an API to develope GUI applications in java.Java AWT components are platform-dependent i.e. components are displayed according to the view of operating system. AWT is heavyweight i.e. its components are using the resources of OS.The java.awt package provides classes for AWT api such as TextField, Label, TextArea, RadioButton, CheckBox, Choice, List etc.

Swing:

Java Swing tutorial is a important part of Java Foundation Classes (JFC) that is used to create window-based applications. It is built on the top of AWT (Abstract Windowing Toolkit) API and entirely written in java. Unlike AWT, Java Swing provides platform-independent and lightweight components. The javax.swing package gives classes for java swing API such as JButton, JTextField, JTextArea, JRadioButton, JCheckbox, JMenu, JColorChooser etc.

Multithreading:

Multithreading in java is a process of runningor executing multiple threads simultaneously. Thread is basically a lightweight sub-process, a smallest unit of processing. Multiprocessing and multithreading, both are used to achieve multitasking. It allows concurrent execution of two or more parts of a program for maximum utilization of CPU. Threads can be created by using two mechanisms :

1. Extending the Thread class
2. Implementing the Runnable Interface

We create a class that extends the java.lang. Thread class

File Handling:

The java.io package contains all class you might ever need in Java. All these streams represent an input source and an output destination. The stream in the java.io package supports many data such as primitives, object, localized characters, etc. A stream can be defined as a sequence of data. There are two kinds of Streams Input-Stream. The Input-Stream is used to read content from a source Output-Stream. The Output-Stream is used for

writing data to a destination. Though there are many classes related to byte streams but the most frequently used classes are, File Input-Stream and File Output-Stream.

LITERATURE REVIEW:

In [1] they propose a low overhead, on-line memory monitoring scheme utilizing a set of novel hardware counters. The counters indicate the marginal gain in cache hits as the size of the cache is increased, which gives the cache miss-rate as a function of cache size. Using the counters, they describe a scheme that enables an accurate estimate of the isolated miss-rates of each process as a function of cache size under the standard LRU replacement policy. This information can be used to schedule jobs and also to part the cache to minimize the overall miss-rate. The data collected by the monitors can also be used by an analytical model of cache and memory behavior to produce a more accurate overall miss-rate for the collection of processes sharing a cache in both time and space. This overall miss-rate can be used to improve scheduling and partitioning schemes.

The [2] Researched in real-time scheduling often assume that the performance of a computing resource does not change over time. However, as system softwares and system architectures become increasingly complex, resource performance degradation over time becomes more evident. In [2], they study the schedulability of a hard real-time task set on a resource which has performance degradation over time with a known pattern and use both cold and warm periodic rejuvenations as countermeasures.

Nowadays, high performance multicore processors implement multithreading capabilities. The processes running parallel 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 they present a process scheduler for SMT multicores 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 L1 cache and main memory. To deal with fairness, they [3] estimates the progress experienced by the processes, and gives priority to the processes with lower accumulated progress. Experimental results on an Intel Xeon E5645 featuring six dual-threaded SMT cores show that the proposed scheduler improves both performance and fairness over two state-of-the-art schedulers and the Linux OS scheduler. Compared to Linux, unfairness is reduced to a half while still improving performance by 5.6 percent.

In this [4] deriving technologies of persistent memory, such as PCM, MRAM, provide opportunities for preserving files in memory. Tradi-tional 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 [4] presents a framework based on a new concept, File Virtual Address Space. A file system, Sustainable In-Memory File System (SIMFS), is designed and implemented, which fully utilizes the memory mapping hardware at the file access path. First,SIMFS inserts the address space of an open file into the process of address space. Then, file accesses are handled by the memory mapping hardware. Several optimization strategies are also presented for the proposed SIMFS. Extensive experiments are conducted.

The [4] results show that the throughput of SIMFS achieves significant performance improvement over the state-of-the-art in-memory file systems.

Existing System :-

In existing system there is problem of proper utilization of main memory because system uses only serial blocks of memory. To overcome this problem propose system use split and concate techniques for data sharing and we also used multithreading. The propose system have restriction is that it should use minimum dual core system. Hardware requirements of propose system are 1 GB RAM and 20 GB Hard Disk. And it used software technologies as java and J2EE. And we used J2SDK1.5 or later version of Java.

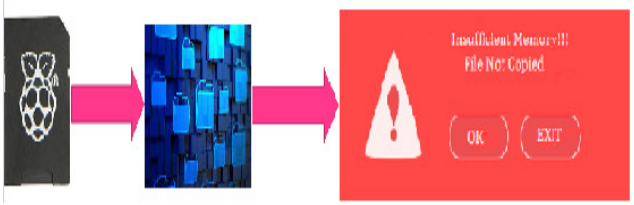

Fig. 1. Example of an existing file system

Proposed System:-

In proposed system we have Memory Management, Space Utilization, Audible Usability, Secure Data Transmission for comfortable use by user.
The Audible Usability is taken into consideration because if user have some different work he can work for it and it will get automatically notified by available audible message i.e beep voice or automated voice message. In Existing System we don't have such Feature of notifying user about his/her work is done or not. To check his work his done or not he has to continuously sit in front of window to notify himself.

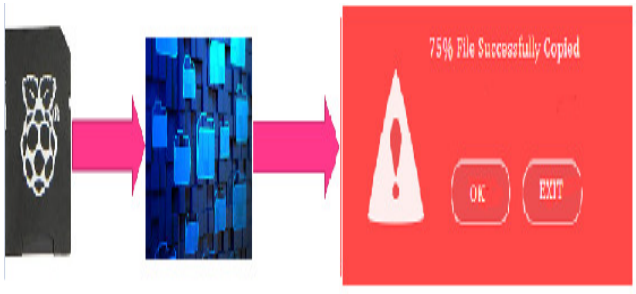

Fig. 2. Example of proposed file system

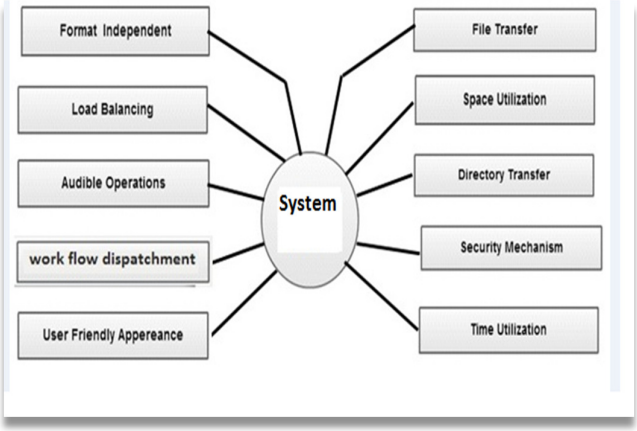

Fig. 3 System Architecture of Proposed System

The System will have all in one features like Format Independent, Load balancing, Audible Operator, User Friendly environment, Space utilization, Security Mechanism, Time utilization

Algorithm:

1. Space Utilization:
1. Select input source.
2. Select output source.
3. if(memory==file_size)
   Transfer file as it is
Else
   Perform file divide operation
4. Transfer file one by one.
5. Select output location.

2. File Concatenate:
1. Select First part.
2. Select Second part.
3. Choose destination path.
4. if(memoryfree)
   Concatenate
Else
   Terminate operation

3. Security Mechanism:
1. Select input file.
2. Select output file.
3. Choose encoding transfer.
4. Insert password for encoding.
5. Transfer file data in encoded form.

Advantages:

1).Portable.
2).Easy To Use
3).Effective Memory Optimization: The system effectively uses available memory space.
4).Faster Data Processing: All the tasks are controlled by single system which leads to fast data processing.

CONCLUSIONS

This survey describes the design of our proposed system. Our approach will be specially designed to handle a number of deletions linear in the length of the file for different operations on IOT system where space utilization , security mechanism ,splitting and concatenation operations are performed on file information. The security will be provided to the file by encrypting it while transferring data which will eventually consume less time.

REFERENCES

[1] A New Memory Monitoring Scheme for Memory-Aware Scheduling
and Partitioning
Authors: G. Edward Suh, Srinivas Devadas, and Larry Rudolph

[2] Xiayu Hua, Chunhui Guo, Hao Wu, Douglas Lautner, and Shangping Ren**,** Schedulability Analysis for Real-Time Task Set on Resource with Performance Degradation and Dual-Level Periodic Rejuvenations

[3] Josue Feliu, Julio Sahuquillo, Member, IEEE, Salvador Petit, Member, IEEE, and Jose Duato, PerfFair: A Progress-Aware Scheduler to Enhance Performance and Fairness in SMT Multicores.

[4] Edwin H.-M. Sha, Senior Member, IEEE, Xianzhang Chen, Qingfeng Zhuge, Member, IEEE, Liang Shi, Member, IEEE, and Weiwen Jiang, A New Design of In-Memory File System Based on File Virtual Address Framework.

[5] STIJN EYERMAN and LIEVEN EECKHOUT Memory-Level Parallelism Aware Fetch Policies for Simultaneous Multithreading Processors.

[6] G. Meletiou etal, Design and Implementation of an e-exam system based on the Android platform",16th Panhellenic Conf. on Informatics, pp.375-380, Oct-2012.

[7]J. Dean and S. Ghemawat, Mapreduce: Simplified data processing on large
clusters, Commun. ACM, vol. 51, no. 1, 2008.

[8]A. Mathur, M. Cao, S. Bhattacharya, A. Dilger, A. Tomas, and L. Vivier, The
new ext4 filesystem: current status and future plans, in Proc. Linux Symp., 2008.

[9]T. Moscibroda and O. Mutlu, Memory performance attacks: denial of memory
service in multi-core systems, in Proc. 16th USENIX Secur. Symp., 2007, pp.
18:118:18

[10]C. D. Antonopoulos, D. S. Nikolopoulos, and T. S. Papatheodorou, Realistic
workload scheduling policies for taming the memory bandwidth bottleneck of
SMPs, in Proc. 11th Int. Conf. High Perform. Comput., 2004, pp. 286296.

[11]C. Luque, M. Moreto, F. J. Cazorla, and M. Valero, Fair CPU time accounting
in CMP+SMT processors, ACM Trans. Archit. Code Optimization, vol. 9, no. 4,
pp. 50:150:25 , Jan. 2013.