Effective Scheduling on Mobile Embedded System

Chieh-jung Hwang

Wei Mu

Ajay Sampat

Table of contents

List of tables/figures

Abstract

The scheduling algorithm is one of the most important portions of embedded operating system. The performance of scheduling algorithm influences the performance of the whole system. We present a Hybrid scheduling algorithm for improving embedded system performance, the algorithm uses standard deviation as the factor to select suitable scheduling algorithm dynamically. The experimental results shows that the hybrid scheduling algorithm can well meet our requirement.

1. Introduction

• Objective
Develop an improved hybrid scheduling algorithm on embedded system and experiment the result.

• What is the problem
Since embedded system has strict requirement on running time and power consumption, reducing turn around time and waiting time is our main propose while minimizing CPU utilization.

• Why this is a project related the this class
Scheduling, CPU utilization, RTOS and performance.

• Why other approach is no good
Other approach do not provide dynamic algorithm change between FCFS and RR,Not been investigated in low power single process mobile systems

• Why you think your approach is better
Compare with others, our approach use hybrid algorithm that can select different algorithm for different task weight to improve the performance.

• Statement of the problem
Our goal is to reduce turn around time and waiting time while minimizing CPU utilization and provide dedicated priority scheduling on RTS

• Area or scope of investigation
Scheduling Algorithms in Mobile Embedded System

2. Theoretical bases and literature review

• Definition of the problem
Like multi-cores have their own market, embedded systems are extensible used these days in mobile systems and Netbooks. Our problem is to guarantee fairness, no starvation and minimizes wait and turnaround time for scheduling processes.

• Theoretical background of the problem

Scheduling problems are as old as computers itself. In computer science, a scheduling algorithm is the method by which threads, processes or data flows are given access to system resources (e.g. processor time, communications bandwidth). This is usually done to load balance a system effectively or achieve a target quality of service. The need for a scheduling algorithm arises from the requirement for most modern systems to perform multitasking (execute more than one process at a time) and multiplexing (transmit multiple flows simultaneously).

• Related research to solve the problem

Wei chose the paper on Modern embedded CPU systems which rely on a growing number of software features, but this growth increases the memory footprint and increases the need for efficient instruction and data caches. The embedded operating system will often juggle a changing set tasks in a round-robin fashion, which inevitably results in cache misses due to conflicts between different tasks. The technique reduces cache misses by continuously monitoring CPU cache misses to grade the performance of running tasks. Through a series of step-wise refinements, the software system tunes the round-robin ordering to find a better temporal sequence for the tasks. The benefits of this technique are illustrated using an ARM processor running application benchmarks with different cache organizations and round-robin scheduling techniques.

Ajay chose the paper with a simple scheduling scheme called Minimized Cycle Round Robin (MCRR) for packet and cell switched networks. Paper demonstrate that MCRR exhibits optimal worst-case latency property in a certain class of weighted round robin (WRR)approaches which are currently used for packet (cell) scheduling in high-speed networks, such as ATM networks. We further present a Hierarchical MCRR (HMCRR) mechanism to maximize the efficiency of MCRR in practice and be scalable to large number of connections.

Chieh-jung chose the paper analyzing the task management and scheduling algorithm of embedded systems, we present an improved embedded system scheduling algorithm and increase of time slice cycle algorithm. The embedded system is improved upon a real-time system, in which assignment priority scheduling is primary and time slice cycle scheduling is secondary.

• Advantage/disadvantage of those research
The embedded operating system will often juggle changing set tasks in a round-robin fashion, which inevitably results in cache misses due to conflicts between different tasks. The technique reduces cache misses by continuously monitoring CPU cache misses to grade the performance of running tasks. Through a series of step-wise refinements, the software system tunes the round-robin ordering to find a better temporal sequence for the tasks. This tuning is done dynamically during program execution and hence can adapt to changes in work load or external input stimulus.

HMCRR can significantly improve router/switch scheduling latency and hence the delay and delay jitter performance, while keeping the simplicity of WRR based scheduling disciplines.

Through the application of collision detection algorithm and path planning algorithm in the system, it shows that the improved embedded system can well meet the requirement, with good usability.

• Your solution to solve this problem
A switch between the FCFS (non preemptive) & RR (preemptive) depending on a standard deviation we tested to be the best switching moment.

• Where your solution different from others
It is an effective Hybrid approach. This will make the best use of RR & FCFS algorithms.

• Why your solution is better
It makes best use of preemption and non preemption algorithms in a mixed model. We use a fixed standard deviation (easily changeable) that can help us find the optimal scheduling mixture.

3. Hypothesis

We claim our Hybrid algorithm delivers better turn around time and waiting time for scheduling on embedded systems(single & multi-core) while also reducing context switches & delay.

Our algorithm guarantees more fairness than the average FCFS or RR alone.

4. Methodology

• How to generate/collect input data
Input data is considered as a list of processes' weight, the list's format is:
Weight: Process weight
The input data file can be generated manually or automatically by a script.

• How to solve the problem
  • algorithm design
    RR/FCFS hybrid
  • language used
    C++/Perl
  • tools used
    Eclipse, Visio, Unix

• Prototype
The prototype is composed of four parts: input data, process analyzer, scheduler and output data. Input data is the processes list for scheduling; process analyzer analyzes the processes and decides which scheduling method will be selected, the selection is sent to scheduler, scheduler executes scheduling action(RR/FCFS/hybrid) and generates processes running information; running information is recorded into output data part.

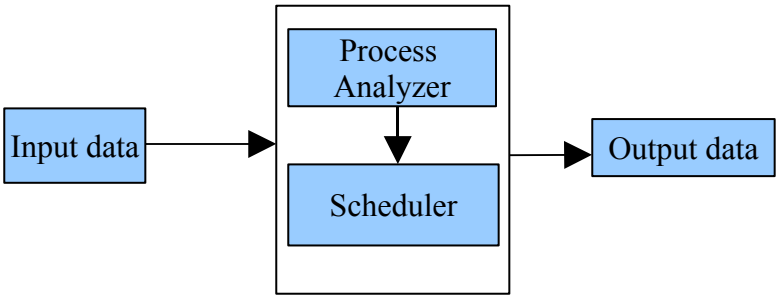


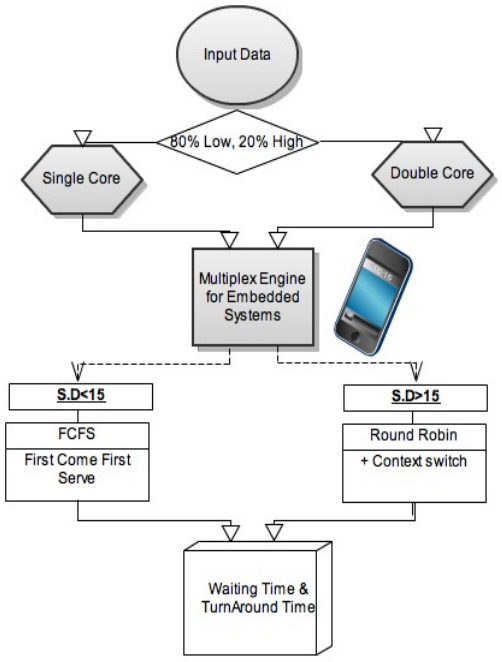
Figure 1. prototype

• How to generate output
Output data is the collection of result, the format is:
Waiting Time/Turn around Time.
Waiting time: process waiting time
Turn around time: process turn around time
After all processes are finished, the output file is generated by scheduler.

• How to test against hypothesis
We run our experiment by following procedure:
Generate input data,
Use our scheduling model to test it with FCFS/RR/hybrid method,
Get the performance comparison among them.

• How to proof correctness
If hybrid scheduling can result in reduction of average turn around time, we can prove our solution is correct.

5. Implementation



6. Data analysis and discussion

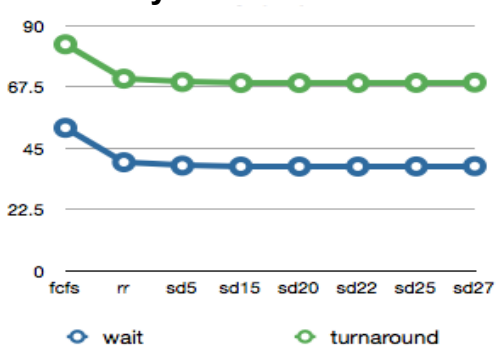


Figure 3: Wait and turnaround time comparison
Y-axis shows CPU waiting time and turnaround time. X-axis shows in pure FCFS and RR algorithm, and the hybrid use Standard Deviation to indicate different algorithm select factor.

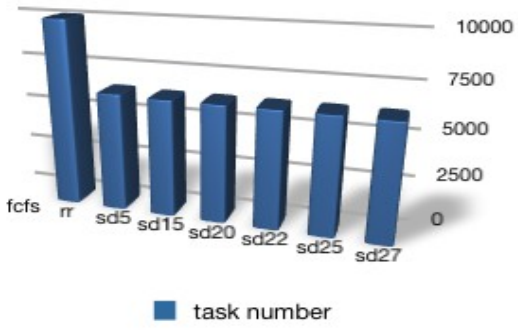


Figure 4: Task number comparison
Y-axis shows different implement. X-axis shows the task number in round robin algorithm of different implementation.

These results show that implementing a hybrid algorithm improves not only waiting time but also turnaround time. Furthermore, increase standard deviation between 5 to 15 also increase the performance. This value is related to the task number, using smaller standard deviation means more tasks using RR but some tasks may be work fine in FCFS. In conclusion which task go RR which task go FCFS play an important role in our implement.

We claimed in our hypothesis to create a hybrid algorithm paper to reduce delay(waiting time) & provide optimal turnaround time for embedded systems. Our Output clearly shows we were able to achieve at least 3% but at most 20% of improved. Since We can see FCFS wastes some time in heavy task, and RR not only wastes some time in lightly task but the time delay. Moreover, when to use RR or FCFS can be configurable, that is, we can adjust different value for different jobs queue. We all know that too small SD or too large SD also work worse. Thus, if using more precise select factor you can get better performance.

7. Conclusions and recommendations

This paper has introduced Hybrid Scheduling as a flexible scheduling algorithm for improving the run-time performance of embedded systems. By using Standard Deviation as the algorithm selection factor, system can select an optimal scheduling algorithm for current tasks. With our limited experimentation, we were able to achieve a 3% to 22.89% improvement in reduction of wait time and 2% - 9.23% improvement in the reduction of turn around time. Our results indicate that Hybrid scheduling has better performance than FCFS and RR. This can be important for multi-tasking embedded system. besides this, the code overhead is low and the implementation is flexible. So, as a fixed scheduling algorithm, RR and FCFS can provide good performance in certain conditions, but not for all conditions. The hybrid technique in contrast, monitor the input tasks and dynamically adjusts the scheduling algorithm to improve performance under changing conditions.

For the future work, priority scheduling, multi-core scheduling and IQR(Interquartile Range) should be

considered.
Due to the limited time, we didn't add priority scheduling in our hybrid scheduling. Priority scheduling is important in some embedded system and real-time system.
Multi-core devices have been established for several years, and they are just the beginning of the explosion in the number of cores per device. However, this kind of expansion creates a challenge for scheduling. We need to know what is the suitable scheduling algorithm for the multi-core system.
We currently use SD(standard deviation) as the algorithm selection factor for selecting suitable algorithm. IQR, a measure of statistical dispersion, could be a better choice for the factor but it is arguable that the code overhead might be heavy.

8. Acknowledgment

We wish to thank Professor Ming-hwa Wang for his guidance and thank our team member's hard work.

9. Bibliography

[1] Yumei, Xiong, Yimin, Chen, An Improved Task Scheduling Algorithm in Embedded Systems, Intelligent Information Technology and Security Informatics (IITSI), 2010 Third International Symposium on ,682 - 685,2010
[2] Batcher, K.W., Walker, R.A., Dynamic Round-Robin Task Scheduling to Reduce Cache Misses for Embedded Systems, Design, Automation and Test in Europe, 2008. , page260 - 263, 2008
[3] Yao Liang, A Simple and Effective Scheduling Mechanism Using Minimized Cycle Round Robin, Communications, 2002. ICC 2002. IEEE International Conference on, page 2384 - 2388 vol.4, 2002
[4] Andrew s. Tannenbaum, Modern Operating System, Third edition