A Weighted-Selective Scheduling Scheme in an Open Software Radio Environment

Kan ZHENG *†, Gang Li † and Lin HUANG*
*Wireless System Lab
France Telecom, R&D Center (Beijing), China
†School of telecommunication engineering
Beijing University of Posts&Telecommunications, Beijing, China

Abstract—Software-defined radio (SDR) is the promising technology that allows the different wireless standards easily be converged. Using general purpose processors and open-source operating systems instead of dedicated hardware and software to build up the open wireless communication system platform is very flexible and low-cost solution. In this paper, an adaptive weighted-selective (WS) scheduling scheme in the open platform consisting of common PCs for real-time application is proposed. Then, it has been demonstrated by the simulations as the efficient distributed strategy for parallel computing in the heterogeneous environment.

key words: software radio; real-time scheduling

I. INTRODUCTION

A software-defined radio (SDR) system is a radio communication system which can tune to any frequency band and receive any modulation across a large frequency spectrum by means of programmable hardware which is controlled by software. Up until now, the dedicated digital signal processing (DSP) processors have been the well-known technology associated with the development of SDR systems. Typical applications involve the digitization of baseband or intermediate frequency (IF) signals, thus allowing software defined hardware to perform the remainder of radio frequency (RF) processing tasks. However, the implementation complexity and cost increases when the digitization signal is as close as possible to the antenna. Meanwhile, the assemble program languages used for DSP development are often difficult to learn and it takes long time for engineers to be able to write the effective codes even sometimes C/C++ language can be used together.

Common PCs continue to be improved rapidly in processor speed and memory size while maintaining outstanding cost/performance. Another novel approach for software radio does rely on general purpose processor such as Intel Pentium instead of the special purpose DSP processors. It can be seen that the cluster of common PCs is a compelling platform for implementing the wireless systems. In such platform, the commonplace language such as C/C++ and the familiar development tools can be used, which means the software radio design is open to a new community of software developers [1][2][3]. Meanwhile, the parallel processing is being seen as the only cost-effective method for the fast solution of computationally large and data-intensive problems[4][5][6].

The main objective of the parallel computing environment design in this paper is to support real-time digital signal processing application on general purpose PC-cluster platform. The implementation of digital signal processing functions is highly delay-sensitive and too much jitter may greatly deteriorate the system performance. To reduce the jitter, the units of the digital signal must be processed timely and orderly in the platform. Thus, when both the computation complexity of each digital signal unit and the real-time requirements are taken into account, the requirements for designing the scheduling algorithm will be: 1) to achieve the high computation ability of the whole PC-cluster platform with the minimal delay; 2) to maintain the flow order of the outgoing signal stream after being processed to reduce the jitter. Therefore, an adaptive real-time scheduling scheme for the software radio platform with common PCs is presented in this paper.

II. SYSTEM OVERVIEW

In this section, we describe the basic architectural characteristics of the wireless PC-cluster platform. The main objective is to implement the wireless system from the physical to upper layers by software in the common PCs. Furthermore, this platform is developed on an open-source operating system such as Linux. The basic hardware/software architecture for PC-cluster platform is shown in Fig.1.

The hardware of the testbed for this PC-cluster platform consists of 2 elements that are under software-control, which include:
- GNU Universal Software Radio Peripheral (USRP) card including RF/IF frontend, analog/digital (A/D) and digital/analog(D/A) convertor
- common PCs

According to the different functionality, the whole software
architecture of this platform comprises five major parts, which includes the upper layer modules, the DSP function modules of physical layer, the controller of USRP card, the user interface modules and parallel computing modules. And this paper mainly focus on the last part.

The main objective of the parallel computing environment design is to support real-time signal processing application on a general purpose PC-cluster platform. First, we discuss how to build up such a parallel computing environment for the software radio implementation.

Each PC can be regarded as one node in the parallel computing environment. Usually the common parallel programming software such as Openmosix and MPI can be used in the parallel computing systems for non-real-time applications. However, the latency and jitter of data transmission between different nodes can’t meet the real-time requirement for digital signal processing. And their task scheduling flexibilities are not good enough for signal processing applications. Therefore, we have to design the parallel computing schemes based on UDP socket programming.

Since the computation complexity of the receiver is much more than that of the transmitter, only the parallel computing environment of the transmitter module is discussed and its architecture is shown in Fig.2. The receiver consists of one distributing node $P_d$, a few worker nodes, $P_1 \text{ through } P_K$, and one collecting node $P_c$. Here $K$ is the number of the worker nodes in this platform. The distributing node $P_d$ has to not only receive the continuous high-speed data samples from the AD convertor, but also extract useful data and distribute them to the worker nodes for digital signal processing. The collecting node $P_c$ receives the packets from processor $P_1 \text{ through } P_K$ and sends them to the upper layers for further processing.

The data to be processed in the PC-cluster platform is the samples generated by AD convertor. To be processed in a distributed computing environment, the data stream should be divided into blocks. As we know, most of data streams that wireless communication systems handle can be separated temporally. It means that two fragments of the data stream could be processed independently if they do not overlap each other. For wireless communication system, the data streams have frame structures and can be processed based on the unit of frame. The length of a frame determines the latency time of the processing, so it cannot be too long. If we separate the data stream in frames, the data blocks will not be too large for processors to handle. On the other hand, the correlation of two frames depends on the interval between them and the coding scheme of the system. In general, the data frames have little correlation between each other when the interval between them is long enough. So the temporal decoupling is feasible for our platform.

After the decoupling in the distributing node, the data will be dispatched to the worker nodes for processing. Each worker node independently processes the frame units using the local computing resource. There is no digital signal processing in the distributing node because the scheduling function will keep this node busy. The main functions of three kinds of the nodes are shortly described in the follows:

A. Distributing node

In this node, the samples of the signal continuously sent from AD convertor will be put into the frame buffer before being distributed to the worker nodes. The simple FIFO policy is applied in the frame buffer since the input samples are time-aligned. The input samples can be accepted until the buffer is full. The distributing manager fetches the frames from the buffer and distribute them to the worker nodes according to the particular scheduling algorithm. When all the worker nodes are busy, the accepted frames will stay in the buffer until the computing resources become available. There is a status table for the scheduling in the distributing node. It also collects the load statistics information and record them in the table if the feedback information is available. The load between different worker nodes can be balanced by the scheduling algorithm.

B. Worker node

The processes for digital signal processing are running in the worker node, which make the necessary processing on the input signal samples of one frame. The processed signal will be sent to the collecting node once the process is finished. Also the status information and load statistics of each worker node will be reported to the distributing node in time. The processing time per frame depends not only on the implementation complexity of signal processing but also on the computation abilities of the worker node. So the number of this DSP process in one work node should be optimized if the multiple processes run in the same worker node.

C. Collecting node

The processed samples from the different worker node will be accepted and stored in the buffer of the collecting node. The different policy of fetching these samples from the buffer depends on the scheduling algorithm which will be described in the next section.

In this parallel computing environment, we will try to minimize the processing time for each frame and to preserve the frame order of the outgoing signal with minimal time delay.
III. SCHEDULE ALGORITHMS FOR DIGITAL SIGNAL PROCESSING

One of the critical issues in implementing this PC-cluster SDR platform is the scheduling algorithms for parallel computing, i.e. how to distribute multiple frames among different worker nodes to achieve the best utilization with the smallest time delay. There are two objectives of the scheduling algorithms: 1) the computation load is well-balanced among the worker nodes $P_1$ through $P_K$; 2) the flow order is maintained in the collecting node $P_e$ after the signals are processed in the worker nodes with minimal time delay. A frame-based round robin scheduling is the simplest but low-efficiency in the heterogeneous computing environment (i.e. the different worker nodes have the different computation abilities). Also, it has poor performance in term of out-of-order degree for the processed frame in case of heterogeneous computing environment. Here we also propose another adaptive scheduling algorithm, which has higher efficiency without out-of-order problem.

A. Schedule algorithms

1) Round Robin Algorithm: With round robin, the distributing manager searches for an available worker node in the fixed order. The status parameter $s_k(n), k = 1, 2, ..., K$ of the worker node can be marked as "0" or "1" for "busy" or "idle" in the $n$th frame interval. In this case, once the distributing manager starts to send a frame unit to a worker node at fixed frame interval $T_s$, the worker node will be labeled as "0" in the status table; and when the computation task is completed and the feedback is successful, this worker node will be labeled "1" again and can be chosen to be worked. The processed samples are delivered to the collecting node and put into the buffer. Then a simple FIFO policy is adopted to fetch the samples for the processing on the upper layers. This scheme is efficient and keep the order only in the homogeneous computing environment. Otherwise, in the heterogeneous computing environment, the processing time for one frame in the worker nodes are different and the processed samples can’t arrive at the collecting with the preserved order.

2) Weighted-Selective Algorithm: In order to let the system work low-latency and efficiently as well as keep the order of the processed samples among the frames, we propose and implement the weighted-selective scheduling algorithm in the parallel computing environment.

Compare to the round robin algorithm, the weights $w_k(n), k = 1, 2, ..., K$ for representing the average computation ability is also introduced for scheduling in the distributing node, which are determined by the computation ability of the worker node and its current working status. When the adaptive scheduling algorithm is adopted in the $n$th frame interval, one frame fetched from the buffer can be distributed to a particular worker node according to the function defined as

$$ l = \arg \max_{k \in \{1, 2, ..., K\}} \frac{s_k(n)}{w_k(n)} $$

(1)

where $l$ is the index of the worker node to which the frame will be distributed for processing. Concerning the weights which describe the processing capacity of each worker node in the status table, the worker node should update and feedback its load information indicator $w_k(n)$ at fixed time interval (e.g. $T_s$) or when it becomes "idle" immediately.

In our platform, we define the load information indicator $w_k(n)$ in the $k$th worker node during the $n$th frame interval as

$$ w_k(n) = L_k(n)/T_s, k = 1, 2, ..., K $$

(2)

where $L_k(n)$ is the CPU time spent by the digital signal processing for one frame signal if the $k$th worker node become "idle" during the $n$th frame interval. Otherwise, this indicator will keep unchanged, i.e. $w_k(n) = w_k(n-1)$.

B. Fetch Policy in the collecting node

In order to avoid the out-of-order problem and frame loss, the fetch policy adopted in the buffer of the collecting node can’t be simple FIFO. The packet containing the processed samples from the worker node to the collecting node should also include the frame order index $i, i = 1, 2, ..., M$, in its head area. On the other hand, there is one counter $j$ in the collecting node that can be increased from 1 to $M$ circularly, where $M$ is the maximum number of the processed frame in the buffer. After the packet of the processed samples is stored into the buffer, its frame order index will be picked up at first and compared with this counter. The detailed fetch policy is shown in Fig.3.

Fig. 3. Flow of fetching scheme in the collecting node

The proposed weighted-selective scheduling algorithm can be summarized as follows: all the worker nodes are initialized as the same load indicators $w_k(n)$ and status indicators $s_k(n)$ at the beginning of the process (i.e. $w_k(n) = s_k(n) = 0; k = 1, 2, ..., K$). Then, the load statistics information will be feedback from the worker nodes to the distributing node at a fixed time interval or evoked by the idle events. The collected load statistics information is calculated to determine the average computation ability weights of different worker nodes in the distributing node. The particular "idle" worker node with the minimal weight will be selected to be the next
processing node for processing digital signals. In the collecting node, the well-designed fetching policy in the buffer is applied in order to avoid the out-of-order problem and frame loss.

IV. EXPERIMENTAL RESULTS AND DISCUSSION

Our experiments are performed on the common PCs with the operating system of Fedora Core 4. Different PCs play the different functions including distributing node, worker node and collecting node. Usually there is only one distributing node and one collecting node in the PC-cluster SDR platform while the number of the worker nodes is dependent on the computation requirement. All the programs including digital signal processing and scheduling algorithms are programmed by C/C++ language.

The baseband receiver based on TD-SCDMA standard is implemented in our PC-Cluster platform as an example. The received signal at the distributing node is divided into the frames with the length of \(40 \, \text{ms}\), i.e. \(T_s = 40 \, \text{ms}\). The computation required by processing one frame of TD-SCDMA signal with 64Kbps data service is regarded as one task unit. And all the tasks are distributed to different worker nodes according to the scheduling algorithm by the distributing node. The heterogeneous computing environment is simulated in order to compare the different scheduling algorithms, i.e. different PC is modeled to have the different computation ability in the different moment. It is assumed that CPU time spent by one task at the \(k\)th worker node is \(p_k(n) = \alpha + \beta g_k^2(n)\), where \(g_k(n)\) is the Gaussian random variable with the mean of \(0\)ms and the variance of \(1\)ms, the constant \(\alpha\) and \(\beta\) are both set to be \(40\)ms. The communication delay and overhead can be omitted due to the high bandwidth between the nodes.

Fig.4(a) gives the frame loss rate (FLR) of the RR and WS schemes without the buffer at the collecting node. It can be seen that the FLR decreases more rapidly when using WS scheme than when using RR scheme. Also, the frame loss can be avoided if number of the worker node is large.

In Fig.4(b), the WS scheme achieves smaller out-of-order ratio than RR scheme because it maintains an in-order delivery pattern and well balances the loads between different worker nodes according to the feedback information. Furthermore, the out-of-order problem can be solved by the buffer with the given fetch policy, which will increase the delay and jitter litter as shown in the followed results.

Fig. 4(a). Frame Loss rate and Out-of-order rate comparison between different scheduling scheme without the buffer

Fig. 4(b). Delay performance comparison between different scheduling schemes

The delay performances of different scheduling algorithms are compared in the Fig.5. Since the Round Robin(RR) scheme only distributes the tasks in the fixed order without considering the heterogeneity of each worker node, the delay and jitter are large and almost not affected by the number of the worker node. On the other hand, with the Weighted-Selective (WS) scheme, the worker node of the strongest computation ability is usually selected. Therefore, the delay of WS scheme is smaller than that of RR scheme and the gain becomes larger with the number of the worker node increases due to more scheduling flexibility. If the buffer with the given fetch policy is adopted by RR or WS scheme, the delay will become a little larger.

V. CONCLUSIONS

The software-defined radio approach based on common PCs and open operating system provides tremendous flexibility for the implementing of wireless communication systems. An weighted-selective (WS) scheduling scheme with the buffer is proposed for this SDR platform in this paper. The experimental results shown that proposed algorithm satisfies both the load balancing and in-order requirement of digital signal processing. Also, the delay of the proposed WS scheduling scheme is smaller than the conventional RR scheme in the heterogenous computing environment.

REFERENCES