Embedded Data Processing Platform Resource Scheduling Research

Qiang Cui[1], Shufen Liu[2], Qifeng Xu[2], and Tie Bao[2(✉)]

[1] System Engineering Research Institute of China State, Beijing, China
[2] College of Computer Science and Technology,
Jilin University, Changchun, China
baotie@jlu.edu.cn

Abstract. When the embedded data processing platform is dispatching resource, it should first estimate the weight of each processing nodes. This paper uses the grey prediction model and the weights of rotation algorithm to schedule resource for embedded data processing platform. And resource scheduling is verified through the system test of the embedded data processing platform.

Keywords: Embedded system · Resources scheduling · Grey forecasting model · Weights of polling

1 Introduction

With the development of technology, embedded technology has been integrated into our life. Medical electronics, intelligent furniture, logistics management, electric power control, even electronic watches, cars, aircraft, satellites and all the devices with digital interfaces and program control are marked by the embedded system [1]. The resource scheduling of embedded platform aims to regulate, measure, analyze and use various resources reasonably and effectively. Effective resource scheduling algorithm can improve operation speed and operation efficiency better. Based on task completion time and resource load balance, literature [2] finds out the optimal resource scheduling scheme through particle swarm optimization algorithm. Literature [3] uses gray-scale prediction model to realize resource scheduling of virtual machine system through network benchmark test and polynomial modeling. Literature [4] constructs a problem based on logic model and applies the algorithm's satisfiability problem to solve the resource scheduling problem. Literature [5] proposes a Delay scheduling algorithm (RFD) based on resource prediction, which is based on the prediction method of resource availability to schedule jobs reasonably. In this paper, we use grey prediction model to predict the load of CPU, calculate the weight according to the prediction results, rearrange the CPU queue and use the weight polling algorithm to realize resource scheduling.

© Springer Nature Switzerland AG 2019
Y. Tang et al. (Eds.): HCC 2018, LNCS 11354, pp. 93–99, 2019.
https://doi.org/10.1007/978-3-030-15127-0_10

2 Research and Design for Resource Scheduling

2.1 Gray-Scale Prediction Model

The gray-scale prediction model is a prediction method for the long term description of the development law of things through a small amount of incomplete information and establishing the grey differential prediction model. The data of grey-scale prediction is the inverse result of the prediction value obtained by the GM (1, 1) model of data generation.

The GM (1, 1) model assumes that the utilization rate of a node's CPU processing node at N continuous time can be denoted as,

$$W^0 = \left(\omega_1^0, \omega_2^0, \omega_3^0 \cdots, \omega_N^0\right) \tag{1}$$

The GM (1, 1) model assumes that the utilization rate of a node's CPU processing node at N continuous time can be denoted as,

$$W^0 = \left(\omega_1^0, \omega_2^0, \omega_3^0 \cdots, \omega_N^0\right) \tag{2}$$

Then the data are added to the data sequentially as follows,

$$\omega_1^1 = \omega_1^0 \tag{3}$$

$$\omega_2^1 = \omega_1^0 + \omega_2^0 \tag{4}$$

$$\vdots$$

$$\omega_N^1 = \omega_1^0 + \omega_2^0 + \cdots \omega_N^0 \tag{5}$$

A new series can be generated by a cumulative addition, and the new series is denoted as follows,

$$W^1 = \left(\omega_1^1, \omega_2^1, \omega_3^1 \cdots, \omega_N^1\right) \tag{6}$$

It also can be denoted simply as following:

$$\omega^{(1)}(i) = \left\{\sum_{j=1}^{i} \omega^0(j) | i = 1, 2 \cdots N\right\} \tag{7}$$

By accumulating, the vibration and fluctuation of raw data can be weakened. Then the latter item is subtracted from the previous item,

$$\Delta\omega^1(i) = \omega^1(i) - \omega^1(i-1) = \omega^0(i) \tag{8}$$

where $i = 1, 2, \cdots \mathrm{N}, \omega^0(0) = 0$. ω^1 satisfies the first order ordinary differential equation,

$$\frac{d\omega^1}{dt} + a\omega^1 = \mu \tag{9}$$

Where μ is the value of development of gray, which is the constant input to system. The initial condition of the differential equation is, when $\mathrm{t} = \mathrm{t}_0$, $\omega^1 = \omega^1(\mathrm{t}_0)$, its solution is,

$$\omega^1(\mathrm{t}) = \left[\omega^1(t_0) - \frac{\mu}{a}\right] e^{-a(t-t_0)} + \frac{\mu}{a} \tag{10}$$

The discrete value of samples that are sampled equidistantly

$$\omega^1(k+1) = \left[\omega^1(1) - \frac{\mu}{a}\right] e^{-ak} + \frac{\mu}{a} \tag{11}$$

For a cumulative sequence, W^1, this paper estimates the constant μ and a by the least square method. Because $\omega^1(1)$ is the initial value, $\omega^1(2)$, $\omega^1(3) \cdots \omega^1(\mathrm{N})$ are substituted for the equation respectively, and the differential is substituted for differential, and the samples are sample equidistantly, $\Delta\mathrm{t} = (\mathrm{t}+1) - \mathrm{t} = 1$. Then it has,

$$\frac{\Delta\omega^1(2)}{\Delta t} = \Delta\omega^1(2) = \omega^1(2) - \omega^1(1) = \omega^0(2) \tag{12}$$

For the same reason,

$$\frac{\Delta\omega^1(N)}{\Delta t} = \omega^0(N) \tag{13}$$

It can be deduced at the same time,

$$\omega^0(2) + a\omega^1(2) = \mu \tag{14}$$

$$\omega^0(3) + a\omega^1(3) = \mu \tag{15}$$

$$\cdots\cdots\cdots\cdots$$

$$\omega^0(N) + a\omega^1(N) = \mu \tag{16}$$

The $\mathrm{ax}^1(\mathrm{i})$ can be moved to the right, and its vector product form is denoted as following,

$$\begin{cases} \omega^0(2) = [-\omega^1(2), 1] \begin{bmatrix} a \\ \mu \end{bmatrix} \\ \omega^0(3) = [-\omega^1(3), 1] \begin{bmatrix} a \\ \mu \end{bmatrix} \\ \cdots\cdots \\ \omega^0(N) = [-\omega^1(N), 1] \begin{bmatrix} a \\ \mu \end{bmatrix} \end{cases} \tag{17}$$

Then the matrix expression is

$$\begin{bmatrix} \omega^0(2) \\ \omega^0(3) \\ \vdots \\ \omega^0(N) \end{bmatrix} = \begin{bmatrix} -\frac{1}{2}[\omega^1(2) + \omega^1(1)] & 1 \\ -\frac{1}{2}[\omega^1(3) + \omega^1(2)] & 1 \\ \vdots & \\ -\frac{1}{2}[\omega^1(N) + \omega^1(N-1)] & 1 \end{bmatrix} \begin{bmatrix} a \\ \mu \end{bmatrix} \tag{18}$$

The three matrices of the upper equation are expressed as W, B, U respectively.

$$\mathrm{W} = \mathrm{BU} \tag{19}$$

The least square estimation is

$$\hat{U} = \begin{bmatrix} \hat{a} \\ \hat{\mu} \end{bmatrix} = \left(B^T B\right)^{-1} B^T W \tag{20}$$

The corresponding equation of time can be obtained by substituting the estimated value â and $\hat{\mu}$ into the equation.

$$\hat{x}^1(k+1) = \left[x^1(1) - \frac{\hat{\mu}}{\hat{a}}\right] e^{-\hat{a}k} + \frac{\hat{u}}{\hat{a}} \tag{21}$$

When k = 1, 2, … , N − 1, the upper equation is a fitting value. When K >= N, the upper equation is the predicted value. It is equivalent to a cumulative fitting value, which is reduced by subtraction, and then the predicted value is obtained.

2.2 Weighted Rounded Robin Algorithm

Owing to the embedded platform of data processing is a number of isomorphic servers in function, and the data processing process, the processing capacity of components is not very different from each other. However, it needs to consider some real-time situations, such as CPU utilization rate, memory utilization rate, network node congestion and so on. Therefore, this paper uses the Weighted Rounded Robin algorithm. Because the processing capability of the data processing platform is the same, and the Weighted Rounded Robin is preferred. But with each dynamic assignment of the task, the real-time state of the processors will change.

First, the received task queue enters the centralized scheduling server, the resource management component in this paper, second, assigns weights to each server in advance according to the real-time status of the processor, and last assigns the task to the task by the corresponding weight value. Therefore, a reasonable determination of the weight is the key to the allocation of resources. By processing the historical data of the nodes, the CPU load rate of the corresponding nodes can be estimated, and each CPU weight queue is determined according to the CPU load rate.

The Weighted Rounded Robin algorithm uses the node weight to represent the processing performance of the node. The algorithm assigns each processing node by the order of weight value and polling scheduling, and the processing nodes with high weight value are able to handle more task requests with lower weight than the weight value. The algorithm process can be described as follows:

This paper sets the processing node in the cluster as N = {N0, N1, … , Nn − 1}, where the weight value of node Ni can be denoted by W(Ni), and i represents the ID of the last selected sever. T (Ni) represents the amount of Ni currently allocated.

$\sum T(N_i)$ represents the amount of tasks that are required to be assigned currently. $\sum W(N_i)$ represents the sum of the weights of the nodes. Then, it has:

$$W(N_i) \Big/ \sum W(N_i) = T(N_i) \Big/ \sum T(N_i) \tag{22}$$

Computation of weights is an important factor in achieving load balancing. When initializing the system, the operator needs to set the initial weight DW (Ni) for each node according to the conditions of each node.

The dynamic weights of the processing nodes are determined by various parameters in the running state, which mainly include CPU resources, memory resources, response time and the number of modules to run. For data processing, CPU utilization and memory utilization are relatively important. In order to express the weight of each factor, a constant coefficient is introduced, ω_i. This paper uses this constant to represent the importance of the influencing factors, where $\sum \omega_i = 1$. Therefore, the weight value formula of each node Ni can be described as:

$$\mathrm{LOAD}(N_i) = \omega_1 \cdot \mathrm{Lcpu}(N_i) + \omega_2 \cdot \mathrm{Lmemory}(N_i) + \omega_3 \cdot \mathrm{Lprocess}(N_i) \tag{23}$$

Where Lf(Ni) represents the amount of load of a node Ni. The upper equation can represent the CPU utilization ratio, memory utilization ratio and process number. The dynamic weight of the resource scheduling component is run periodically through the state of the program, and the final weight of the system can be calculated through the node weight.

$$W_i = A * DW(N_i) + B * (\mathrm{LOAD}(N_i) - DW(N_i)) * 1/3 \tag{24}$$

In this formula, if the dynamic weight is just equal to the initial weight value and the final weight is unchanged, the load state of the system is just in the ideal state, equal to the initial state DW (Ni). In the case of the constant number of nodes, the task is not dispatched, or the task with low priority is unloaded, and then the task is sent.

3 Validation of Resource Scheduling System

After completing the design and implementation of ship resource scheduling components, we need to carry out a comprehensive system testing of resource scheduling components. In the process of development, each module is unit tested. In this section, the black box testing method is mainly used to verify the functional requirements and non-functional requirements proposed in the requirements.

3.1 Resource Allocation Test

Resource generation is divided into two steps, load forecasting and resource generation. Load prediction verification: First of all, the normal CPU queue is used for load forecasting, weight calculation, and CPU queue is rearranged by weight size. Resource generation: The CPU queue is configured in turn according to the task, and the resource scheduling plan is displayed in the table below (Table 1). However, as a ship display control terminal component, resource scheduling components retain the ability of manual configuration by operators simultaneously. The configuration information is stored in the message queue at this time.

Table 1. The error of fitting of each time.

Time	1	2	3	4	5
Error (%)	0	1.37310	−1.85130	−0.45378	0.87294

3.2 Resource Distribution Test

When the device clicks on the right key, it can be successfully sent to the resource scheme. The embedded data processing platform framework gives the resource scheduling component feedback message according to the load condition after receiving the message of the load message. At this point, the resource scheduling component will prompt the operator, and the operator needs to choose whether to continue executing the scheduling. If necessary, the resource scheduling component will continue to load. If it is not needed, the queue will be emptied and the components loaded will be unloaded.

3.3 Test of Gray-Scale Prediction Module

Embedded internal data platform collects real-time CPU state information in real time, and predict the data through AR model. The prediction curve is shown as follows, Fitting error value of CPU is shown as following:

The parameters, $a = -0.00917$, $\mu = 42.76613$, where the predictive values of time 6, 7, 8 are shown in Table 2.

By predicting the predicted value of CPU load balance, the lower value is the actual observed CPU value. It can be seen from the table that the actual value is not much different from the predicted value, so the application of the grey prediction model to the resource scheduling system has a good prediction result.

Table 2. Grey model predicted value and actual value.

Time	6	7	8
Predictive value	45.01805	45.43289	45.85162
Actual value	45	44	46

4 Conclusion

In this paper, the grey prediction model and the Weighted Rounded Robin algorithm are used to schedule the embedded data processing platform. In the system, the gray prediction model is used to predict the load of the normal CPU queue, and the weight value is calculated. The weight value is used to rearrange the CPU queue, and the weights are polled according to the load message condition. Scheduling algorithm is used for resource scheduling. The constant coefficient is introduced in the Weighted Rounded Robin algorithm, which indicates the importance of the influence factors, thus making the system priority scheduling to important resources and improving the efficiency of the system. Through the resource scheduling of the component testing system and the collection of CPU state information, the prediction value is compared. The results show that the grey prediction model is applied to the resource scheduling system with better prediction results.

References

1. Furuichi, T., Yamada, K.: Next generation of embedded system on cloud computing. Procedia Comput. Sci. **35**, 1605–1614 (2014)
2. He, P., Wu, L., Song, K., Cao, Y.: Resource scheduling in embedded cloud computing based on particle swarm optimization algorithm. Electron. Des. Eng. **22**(10), 88–90 (2014)
3. Deng, X.: The resources of the virtual system dynamic allocation policy. J. Hangzhou Dianzi Univ. **4**, 39–42 (2013)
4. Gorbenko, A., Popov, V.: Task-resource scheduling problem. Int. J. Autom. Comput. **9**(4), 429–441 (2012)
5. Zhang, J.: Embedded Software Designs. Xian University of Electronic Science and Technology Press, Xian (2008)
6. Zolfaghar, K., Aghaie, A.: A syntactical approach for interpersonal trust prediction in social web applications: combining contextual and structural data. Knowl. Based Syst. **26**, 93–102 (2012)
7. Caarls, W.: Skeletons and asynchronous RPC for embedded data and task parallel image processing. IEICE Trans. Inf. Syst. **89**(7), 2036–2043 (2006)
8. Li, H., Luo, L., Zhou, Y.: Design of radar data processing software based on Ruihua embedded real-time operating system. Fire Control Radar Technol. **2018**(1), 18–26 (2018)
9. Park, H., Choi, K.: Adaptively weighted round-robin arbitration for equality of service in a many-core network-on-chip. Comput. Digit. Tech. IET **10**(1), 37–44 (2016)
10. Ravi, S., Kocher, P.: Security as a new dimension in embedded system design. In: Design Automation Conference, pp. 753–760 (2004)