Online Decentralized Adaptive Optimal Controller Design of CPU Utilization for Distributed Real-Time Embedded Systems

Jianguo Yao, Xue Liu, IEEE Member, Xi Chen, Xiaorui Wang, IEEE Member, and Jian Li

Abstract—In large-scale Distributed Real-time Embedded (DRE) systems, the end-to-end tasks contain chains of subtasks distributed on a large number of CPUs. Controlling their CPU utilizations at desired values is one of the most effective ways to ensure system end-to-end deadlines. For these DRE systems, decentralized control is desired to ensure system scalability and global stability. Recently, researchers have proposed solutions based on Model Predictive Control (MPC) for the decentralized utilization control problem. Although these approaches can handle a limited range of execution time estimation errors, the underlying DRE systems may suffer performance deterioration or even become unstable when large estimation errors exist in real systems. In this paper, we propose a new decentralized optimal controller design for CPU utilization to address this problem. The approach leverages Recursive Least Square (RLS) for adaptive model identification and uses Linear Quadratic (LQ) optimal controller for online tasks’ execution rates adjustment. Simulation results demonstrate the proposed approach can ensure good system performance even when large constant or varying execution time estimation errors exist.

I. INTRODUCTION AND RELATED WORK

Distributed Real-time Embedded (DRE) systems have become increasingly important in many applications and attracted much attention recently. Examples of such systems include avionics mission computing, autonomous aerial surveillance, and disaster recovery systems. These systems execute tasks and deliver services conforming to temporal constraints [2]. In order to guarantee the temporal constrains in these DRE systems, one of the most effective methods is to control the CPU utilization under the schedulability bounds on individual nodes [1]. Traditionally, the effectiveness of these approaches depends on the accuracies of the system model and workload.

Because of the open and unpredictable environment of typical DRE systems, the execution times of tasks usually suffer large uncertainties. A key challenge to these open DRE systems is to provide real-time guarantees even when the workload cannot be accurately characterized a priori. Using the worst-case execution time estimation for system design works, but this method may cause system resources waste because a task’s WCET is often much larger than the actual execution time. A scheduling method that can both satisfy the real-time constrains and make full use of resources under the unpredictable environment is highly desired [4]. In the past few years, researchers began to apply feedback control techniques to solve this challenging problem. Lu et al [5] presented a survey of feedback control techniques to ensure real-time guarantee under unknown execution time. Lin et al applied feedback control techniques to real-time scheduling when facing uncertain workloads in [6]. Fu et al [12] designed a distributed utilization feedback controller to handle system dynamics caused by the load balancing for large-scale server clusters. Some works extended the feedback scheduling to the DRE system. e.g. Stankovic et al [7] leveraged feedback control in distributed real-time systems. In these control-theoretic approaches, system CPU utilization is maintained under the CPU schedulability bound through dynamic resource allocation in response to load variations[1][5][10], CPU utilization control target is usually set near the schedulability bounds so as to fully utilize the CPU resources and at the same time guaranteeing real-time deadlines [8][9].

A key challenging problem for the feedback control algorithm in the DRE systems lies in the unpredictable environment and the large workload variations, which may bring difficulty in precise model estimation or in effective control algorithm design. All the above approaches based on feedback control require accurate models of the real-time system or the DRE system, which may be difficult to obtain in realistic systems. Recent work for utilization control based on Model Predictive Control (MPC) was capable of handling execution time variations within a certain range. But this method fails to maintain the system stability and robustness in large workload variations. Detailed discussions on Model Predictive Control scheme in utilization control for real-time embedded systems were presented in [1][3].

In order to solve the challenging problem and provide better control performance for DRE systems under large uncertainties, a new CPU utilization control approach using
Recursive Least Square (RLS) based model identification and Linear Quadratic (LQ) optimal controller is presented in [14]. This approach can successfully adjust to large execution time uncertainties and workload variations. It is worth noticing that this pilot study only addresses the centralized control problem. With the scale of DRE system turning large, the decentralized control algorithm is highly desired to provide better scalability with less computation cost [3]. In this paper, we propose a decentralized RLS-based LQ optimal control approach for DRE systems based on the decomposition scheme discussed in [3]. The main contribution of this paper can be concluded as follows:

1) A new decentralized control algorithm is proposed based on RLS and LQ optimal control. This approach requires communication among localized controllers, and the Feedforward control can easily compensate the couplings of subsystems.

2) The proposed decentralized control algorithm can guarantee the global stability and flexibility of the DRE systems even when large execution time estimation errors exist.

3) The comparisons between the centralized and the decentralized control are conducted. The centralized control has better performance, but with higher computational cost. The centralized algorithm’s computational time consumption explodes as the scale of the DRE systems increases. Hence in the large-scale DRE system, decentralized control algorithm will outperform compared with the centralized control.

The remainder of this paper is organized as follows. Section II presents the problem statement and our solution architecture on CPU utilization control. Section III presents the detailed descriptions of the control system design based on RLS model identification and LQ optimal controller. Section IV presents the simulation and evaluation results. Finally, conclusion of the paper is given in Section V.

II. PROBLEM STATEMENT AND SOLUTION

A. Model Formulation

We consider the same system model as [3]. The application is a group of end-to-end periodic tasks. Each task is composed of a chain of subtasks. A subtask can be released only after its predecessor subtask finishes execution. All the subtasks of one periodic task share the same rate. The hardware platform is composed of multiple processors. Each subtask can be allocated on any processor.

Each task’s rate can be dynamically adjusted within a certain range. The higher rate one task selects the better performance the system provides. After assigning all subtasks on processors, each processor runs a group of periodic subtasks. According to which scheduling policy the processor uses, utilization bound can be computed out to guarantee the real-time constrains.

Each subtask has an estimated execution time. The actual execution time may be significantly different from this estimated value, which causes the utilization variations of this system. The objective of the feedback control is to keep the stability of each processor’s utilization by adjusting task rate. In a realistic system, task execution time may suffer significant variation at runtime, which will cause great variation in the system utilization. The objective of feedback control is to maintain each processor’s utilization at a desired set point by adjusting task execution rates dynamically, assuming that each task’s execution rate can be adjusted within a certain acceptable range for a given application.

The utilization model of processors is defined as follows [3]:

\[ y(k + 1) = y(k) + B \Delta r(k) \]

where \( y \in \mathbb{R}^n \) represents the processor utilization vector on the n nodes; \( \Delta r \in \mathbb{R}^m \) represents the change vector to task execution rates for the m tasks running in the system; \( B \in \mathbb{R}^{n \times m} \) is the unknown input matrix, which is related to parameters of estimated execution time and its corresponding gain \( B \) is described as:

\[ B = GF, \]

where \( G = \text{diag}(g_1, g_2, \ldots, g_n) \) is a diagonal matrix, where \( g_i, i = 1, 2, \ldots, n \) is the scalar value that denotes the ratio between the change to the actual processor utilization and its estimation on processor i. \( F \) is the available subtask allocation matrix that denotes the matching information between each subtask and its corresponding processor. The magnitude of \( g_i \) represents the estimation error, i.e., how much the actual each task’s execution time on processor i deviates from the estimated value, assuming all tasks on processor i have the same estimation error. For example, if \( g_i = 2 \), then each task’s actual execution time is twice the estimated value due to the system uncertainties. We will later show that our controller can handle larger estimation errors (larger values of \( g_i \)) compared with the related work [3] using MPC-based controller in [3] under both the steady workload with fixed \( g_i \) and the varying workload with dynamically changing \( g_i \) at runtime.

Example: Consider a distributed multitasking system from [3], with five processors and six periodic tasks, as shown in Fig. 2. The system model is:

\[
F = \begin{bmatrix}
  c_{11} & 0 & 0 & 0 & c_{13} & 0 \\
  c_{12} & c_{22} & 0 & 0 & 0 & 0 \\
  0 & c_{31} & c_{31} & 0 & 0 & 0 \\
  0 & 0 & c_{42} & 0 & 0 & 0 \\
  0 & 0 & c_{53} & c_{42} & 0 & 0 \\
\end{bmatrix},
\]

\[
G = \begin{bmatrix}
  g_1 & 0 & 0 & 0 & 0 \\
  0 & g_2 & 0 & 0 & 0 \\
  0 & 0 & g_3 & 0 & 0 \\
  0 & 0 & 0 & g_4 & 0 \\
  0 & 0 & 0 & 0 & g_5 \\
\end{bmatrix}
\]

\[
y(k) = \begin{bmatrix} y_1(k) & y_2(k) & y_3(k) & y_4(k) & y_5(k) \end{bmatrix}
\]

\[
\Delta r(k) = \begin{bmatrix} \Delta r_1(k) & \Delta r_2(k) & \Delta r_3(k) & \Delta r_4(k) & \Delta r_5(k) \end{bmatrix}
\]

The control objective is to track the processor utilization target vector \( y_{\text{ref}} \) to minimize the error between the measured processor utilization \( y(k) \) and target utilization \( e(k) = y(k) - y_{\text{ref}}(k) \). The controller task can be implemented either on a separate processor, or on a shared processor with other tasks at the
precondition that it should be assigned the highest priority. There are a utilization monitor and a rate modulator on each processor. The utilization monitor is used to compute the corresponding processor’s utilization in each sampling period, while the rate modulator is used to adjust the tasks’ rates. The controller inputs are the processors’ utilization vector at current sampling time, the utilization target vector and the rate adjustment range of each task. The output of the controller is the change vector to tasks’ rates 
\[ \Delta r(k) = [\Delta r_1(k) \cdots \Delta r_m(k)]^T. \] These outputs are sent to the rate modulators to adjust the corresponding tasks’ rates according to 
\[ r_i(k) = r_i(k-1) + \Delta r_i(k), \quad i = 1, 2, \cdots, m. \] The above description is used in the centralized control architecture and similar decentralized control loop will be given in the next two subsections.

\[ \text{Fig. 1. Decentralized Controller based on RLS and LQ optimal control} \]

B. Model Decomposition

The decomposition concept is to divide the control problem into a set of localized subproblems based on the coupling structure of the whole system. In this paper, we adopt a decomposition approach proposed in [3].

In a set of processors connected with one local controller \( C_i \), there are three types of subtasks, including local subtasks on host processor \( P_i \) (processor controlled by \( C_i \)), neighbor subtasks on \( P_i \)'s neighbors, and all other subtasks in the set of processors. The inputs of the local controller are subtasks’ rates on host processor \( P_i \), neighbor subtasks’ rates on \( P_i \)'s neighbors.

The local model of \( M_i \) is given by
\[ y_i'((k+1)) = y_i'(k) + B' \Delta y_i'(k), \]  
where 
\[ y_i'(k) = [y_i'(k) y_i''(k) \cdots y_i''(k)]^T \] is the CPU utilization vector which consists of \( p \) processors connected with controller \( C_i \), and \( \Delta y'(k) \in R^p \) is the vector of changes to tasks’ rates on all the \( p \) processors.

C. Problem Formulation and Solution

As presented in [14], to handle the model uncertainty due to large variations in execution times, we will apply Recursive Least Square (RLS) based model identification and Linear Quadratic (LQ) optimal controller to control the CPU utilization in the paper. In this architecture, we employ the RLS based model identification method to estimate the underlying computing system online, and use the LQ optimal controller to adjust the tasks’ rates based on the estimated model. Different from the centralized control architecture in [14], in this paper, we apply the controllers design to each subsystem decomposed by subsection B. In each subsystem, the task rates are distinguished as \textit{Selected control inputs} and \textit{Not selected control inputs}. The \textit{Selected control inputs} are tuned using the rate modulators to change the CPU utilizations connected with the corresponding controller. While the \textit{Not selected control inputs} cannot be adjusted in the local subsystem. For example in Controller 3, the task rates of \( T_4 \) and \( T_6 \) are considered as \textit{Selected control inputs}, while the task rates of \( T_3 \) is \textit{Not selected control input}. For the RLS based identification modular in each controller, the measurements of CPU utilizations and the entire tasks’ rates located in the corresponding subsystem are required as inputs of this identification modular. The CPU utilization can be measured by the utilization monitor in the subsystem. Not all the tasks’ rates are available because only the task rates considered as \textit{Selected control inputs} are recorded in the last sample step by the local controller. Communications are required to share the task rates information among the neighbors. After getting \textit{Not selected control inputs}, a Feedforward control is employed to compensate the effects for these task rates. Details of the algorithm will be given in Section III.

Fig. 2 shows the general control system architecture for decomposed subsystem based on RLS modification and LQ optimal controller. Specifically, RLS is used to identify the computing system model, and LQ Controller regulates the system outputs through tuning the inputs. The communication among subsystems shares the information of \textit{Not selected control inputs}. The Feedforward control will compensate the effects of \textit{Not selected control inputs}.

\[ \text{Fig. 2. A general control system architecture for computing system based on RLS-based LQ controller} \]

III. DECENTRALIZED ONLINE ADAPTIVE OPTIMAL CONTROLLER DESIGN

In this section, we describe decentralized online optimal control by using the RLS-based LQ optimal control approach to ensure the CPU utilization. The controller consists of two Modules: a model identification Module to estimate each subsystem online adaptively, and a LQ optimal controller combined with Feedforward control computes the optimal inputs based on the estimated subsystem model.
A. RLS-based Model Identification for Each Subsystem

As presented in Section II, the CPU utilization system can be modeled using a general Multiple Input Multiple Output (MIMO) model [15] described as follows
\[
A'(q^{-1})y'(k) = B'(q^{-1})\Delta sr'(k) + e(k),
\]
where \(A'(q^{-1})\) and \(B'(q^{-1})\) are matrix polynomials in the backward-shift operator
\[
A'(q^{-1}) = I - A_1q^{-1} - \cdots - A_lq^{-l},
B'(q^{-1}) = B_0 - B_1q^{-1} - \cdots - B_lq^{-l},
\]
and the order of the system is \(l\). Due to the uncertainty of the gain matrix \(G\) as described in Section II, we cannot get the input matrix \(B\) accurately. Now we use RLS estimator [15] with exponential forgetting to identify the parameter matrices \(A_i\) and \(B_i\), where \(0 < k \leq l\) and \(0 \leq j < l\). We should use RLS method to identify the model online, since the values of the model parameters may vary in a typical computing service caused by changes in system operating conditions and workload dynamics.

For notational convenience, we rewrite the model in the following RLS-friendly form, which we use in the remainder of the paper
\[
y'(k + 1) = X'(k)\phi'(k) + e'(k + 1),
\]
where
\[
\phi'(k) = [\Delta r'_1(k), \ldots, \Delta r'_{(1-l)}(k-l+1)],
X'(k) = [B'_1, \ldots, B'_l, A'_1, \ldots, A'_l],
\]
RLS estimator with exponential forgetting can identify the time varying parameter matrix \(X'(k)\) online. The estimator has been applied extensively in adaptive control system design since it can converge fast and reject disturbance effectively. The estimator is described by the following equations
\[
\begin{align*}
e'(k + 1) & = y'(k + 1) - \hat{X}'(k)\phi'(k), \\
\hat{X}'(k + 1) & = \hat{X}'(k) + \frac{e'(k + 1)\phi'(k)P(k - 1)}{\hat{\lambda} + (\phi'(k)P(k - 1)\phi'(k))}, \\
P^{-1}(k) & = P^{-1}(k - 1) + P^{-1}(k - 1)\frac{\phi'(k)\phi'(k)}{(\lambda + (\phi'(k)P(k - 1)\phi'(k)))}, \\
\phi'(k) & = (\phi'(k)\phi'(k))^{-1}.
\end{align*}
\]
where \(\hat{X}'(k)\) is the estimation of the \(X'(k)\), \(e'(k + 1)\) is the estimation error vector, \(P(k)\) is the covariance matrix, and \(\lambda\) is the forgetting factor (\(0 < \lambda < 1\)).

B. Linear Quadratic (LQ) Optimal Control for Each Subsystem

After identifying the local model \(M_i\), we apply LQ optimal control [15]. While different from the centralized control problem, there are Selected control inputs and Not selected control inputs for subsystem in the decentralized control. For deduction convenience, we rewrite the model as follows
\[
y'(k + 1) = y'(k) + B'_0\Delta sr'(k) + B'_0\Delta s_t'(k),
\]
where \(\Delta sr'(k) \in R^l\) is the change of task rate which is Not selected control inputs, and \(\Delta sr'(k) \in R^{l-f}\) is the change of task rate which is selected control inputs. \(B'_0 \in R^{m \times f}\), \(B'_0 \in R^{m \times (l-f)}\) are sub-matrix of \(B'_0\) where \(B'_0 = [B'_{10}, B'_{20}]\).

After estimating the system model, we apply LQ optimal controller to the system by minimizing the quadratic cost function as follows:
\[
J^i = \left\| W(y'(k + 1) - y'_r(k + 1)) \right\|^2 + \left\| Q(\Delta sr'(k) - \Delta s_r'(k - 1)) \right\|^2,
\]
where \(W\) is a positive-semi-definite weighting matrix on the tracking errors, and a higher weight implies more importance to the according outputs compared with others. \(Q\) is a positive-definite weighting matrix that penalizes the control variables to minimize the changes in control inputs.

For convenience, we omit the effects of Not selected control inputs item \(B'_0\Delta sr'(k)\) firstly. Then according to derivations of [15], we get
\[
\begin{align*}
\Delta sr'(k) & = ((W\hat{B}'_0)^T W + Q^T)Q^{-1}(W\hat{B}'_0)^T W \\
y_r'(k + 1) - \hat{X}(k)\phi(k) + Q^T Q(\Delta sr'(k - 1))
\end{align*}
\]
where
\[
\phi(k) = [0, \Delta sr'(k - 1), \ldots, \Delta sr'(k - l + 1), \ldots, (y'_r(k - 1), \ldots, y'(k - l + 1)].
\]
However, the effects of Not selected control inputs item may exert large influence on the CPU utilizations located in the subsystem. We adopt the Feedforward control to compensate the disturbance from the Not selected control input, so the control law consists two parts \(\Delta sr'(k) = LQ(k) + FF(k)\), where \(LQ(k)\) is the optimal LQ control part and \(FF(k)\) is the Feedforward control part. The modification of the control law is as follows
\[
\begin{align*}
LQ(k) & = ((W\hat{B}'_0)^T W + Q^T)Q^{-1}(W\hat{B}'_0)^T W \\
y_r'(k + 1) - \hat{X}(k)\phi(k) + Q^T Q(\Delta sr'(k - 1))
\end{align*}
\]
\[
FF(k) = -(\hat{B}'_0)^T \hat{B}_0\Delta sr'(k - 1),
\]

IV. SIMULATION AND EVALUATION

We conducted experimental simulation studies to evaluate the performance of decentralized RLS-based LQ optimal controller for CPU utilization control. In order to show the effectiveness of our approach, we compared the control performance among centralized or decentralized RLS-based LQ controllers and MPC-based controllers. The results are summarized in this section.

A. Simulator and Experimental Setup

The simulation is conducted in Matlab. The simulator consists of three major modules. The experiments are run on a PC with 2.0GHz Pentium dual-core processor and 2.0GB RAM.

286
Supposing $r_i(k)$ is the invocation rate of Task $i$ in the $(k+1)$th sampling period, it has constraints as follows:

$$R_{\text{min},i} \leq r_i(k) \leq R_{\text{max},i}, i = 1, 2, 3, 4, 5, 6,$$

where $R_{\text{min},i}$ is the minimum task rate of task $i$ and $R_{\text{max},i}$ is the maximum.

The bound of CPU utilization is chosen based on the method suggested in [1]. Supposing $x_{\text{max},i}$ is the maximum value of utilization in CPU $i$. It can be calculated by the following equation:

$$x_{\text{max},i} = m_i (2^{m_i} - 1), i = 1, 2, 3, 4, 5, 6,$$

where $m_i$ is the number of subtasks in CPU $i$, and the utilization is subject to,

$$0 < x_i(k) < x_{\text{max},i}.$$

We set the gain matrix $G$ to imply whether the model uncertainties exist or not. In MPC design, we assume gain matrix $G = gI$, $g = 1$, i.e. the actual utilization will be the same as the estimated ones and there is no model uncertainty.

### B. Experiment Results

To compare the system performance with the decentralized and centralized MPC-based controller and RLS-based LQ optimal controller, we conduct three experiments. In the Experiment 1, we choose the small constant workloads uncertainty situation by setting $g = 2$, in which MPC-based controller is stable. In the Experiment 2, we choose the large constant workloads uncertainty situation by setting $g = 7$ in which situation the MPC-based controller is unstable. In the Experiment 3, we set $g_1 = g_2 = 1$ at simulation start time, set them to $g_1 = g_2 = 2$ at the 400th sample step, and finally set $g_1 = g_2 = 0.5$ at the 800th sample step.

Fig. 3-5 (a) and (c) show the processor utilization responses of the centralized and decentralized RLS-based LQ optimal controller, the processor utilization initially jump to 1 due to inaccuracy of model identification, but after a few sampling steps, model identification becomes more accurate, and the task execution rates are adjusted until the processor utilizations converge to the desired set points.

Both of the approaches have acceptable performance, the curves of centralized RLS based LQ optimal controller are a little smoother in the stable region. This is confirmed by Table 1-3, However, the merits for the decentralized control will be discussed in the next subsection.

<table>
<thead>
<tr>
<th>Method</th>
<th>RLS based LQ control</th>
<th>MPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU1</td>
<td>0.0151</td>
<td>0.0166</td>
</tr>
<tr>
<td>CPU2</td>
<td>0.0157</td>
<td>0.0178</td>
</tr>
<tr>
<td>CPU3</td>
<td>0.0153</td>
<td>0.0172</td>
</tr>
<tr>
<td>CPU4</td>
<td>0.0111</td>
<td>0.0134</td>
</tr>
<tr>
<td>CPU5</td>
<td>0.0118</td>
<td>0.0124</td>
</tr>
</tbody>
</table>

Table 2. Aggregate errors in Experiment 2

<table>
<thead>
<tr>
<th>Method</th>
<th>RLS based LQ control</th>
<th>MPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU1</td>
<td>0.174</td>
<td>0.0212</td>
</tr>
<tr>
<td>CPU2</td>
<td>0.189</td>
<td>0.0239</td>
</tr>
<tr>
<td>CPU3</td>
<td>0.187</td>
<td>0.0231</td>
</tr>
<tr>
<td>CPU4</td>
<td>0.0198</td>
<td>0.0298</td>
</tr>
<tr>
<td>CPU5</td>
<td>0.0131</td>
<td>0.0212</td>
</tr>
</tbody>
</table>

Table 3. Aggregate errors in Experiment 3

<table>
<thead>
<tr>
<th>Method</th>
<th>RLS based LQ control</th>
<th>MPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU1</td>
<td>0.0374</td>
<td>0.0412</td>
</tr>
<tr>
<td>CPU2</td>
<td>0.0345</td>
<td>0.0413</td>
</tr>
<tr>
<td>CPU3</td>
<td>0.0382</td>
<td>0.0504</td>
</tr>
<tr>
<td>CPU4</td>
<td>0.0341</td>
<td>0.0475</td>
</tr>
<tr>
<td>CPU5</td>
<td>0.0359</td>
<td>0.0497</td>
</tr>
</tbody>
</table>

Fig. 3-5 (b) and (d) show the processor utilization responses of the centralized and decentralized MPC-based controller. For the Experiment 1 and 3, the processors are initially over utilized, and the task execution rates are then adjusted to converge to the desired utilization set points. For the Experiment 2, the control error falls outside of the stability range of the centralized and decentralized MPC-based controller, and utilizations of both processors oscillate significantly during the entire simulation time. Similar with RLS-based LQ optimal controller, decentralized MPC-based controller has a little worse performance, but it is also acceptable.

![Fig. 3. Processor utilization response in Experiment 1: (a). Centralized RLS based LQ optimal controller; (b). Centralized MPC-based controller; (c) Decentralized RLS based LQ optimal controller; (d) Decentralized MPC-based controller;](image-url)
In this paper, we focus on the decentralized utilization control problem in Distributed Real-time Embedded (DRE) systems. To handle large variations in execution time in the DRE systems, an adaptive control approach is needed to keep the processor utilization at a given set point to ensure schedulability. MPC-based controller can reduce execution time estimation errors within a certain range. To deal with large estimation errors, we employ Recursive Least Squares (RLS) based model identification to estimate the system model online, and then apply Linear Quadratic (LQ) optimal controller to keep the processor utilization at the desired set point by tuning the control inputs. Simulation results show that the decentralized RLS based LQ optimal controller can ensure better system performance than the decentralized MPC-based controller in both constant and varying workload situations.

V. CONCLUSIONS

REFERENCES


