Available online at www.sciencedirect.com
ScienceDirect
IFAC PapersOnline CONFERENCE PAPER ARCHIVE
IFAC-PapersOnLine 49-11 (2016) 685–690

A Parallelized Method for Continuous-Time Models with Dependence on Calculation Order

Kota Sata*, Shun-ichi Azuma**
Akira Ohata***

**Toyota Motor Corporation, Advanced Unit Management System Development Div., Shizuoka, Japan (Tel: +81-55-997-9483; e-mail:kota_sata@mail.toyota.co.jp).*
*** Kyoto University, Dept. of Systems Science, Kyoto, Japan (e-mail:sazuma@i.kyoto.ac.jp)*
**** Technova Inc., Tokyo, Japan (e-mail: ohata@technova.co.jp)*

Abstract: The advancement of the engine control increases the amount of computation. The production ECU (Electronic Control Unit), which is made of single-core architecture, cannot have a higher clock speed. Using multi- / many-core architecture is the only way to decrease execution time. However, when implementing the engine control software, various problems occur in utilization of the multi- / many-core ECU. One of the biggest problems is sequential structure of control software because the software can only execute with one core on the multi- / many-core ECU.

The purpose of this paper is to describe the parallelized control design method, which has decomposed sequential structure and decreases execution time in the embedded multi- / many-core production ECU.

Keywords: automotive control, control design, execution times, electronic control units, internal combustion engine, parallel algorithms

1. INTRODUCTION

Engine controllers become complicated exponentially with the addition of various devices such as the turbocharger for exhaust emission and fuel consumption reduction, increasing the amount of computation. Therefore, the production ECU (Electronic Control Unit) has been improved. Recently, the production ECU has adopted a multi-core CPU.

The improvement of calculation speed of CPUs (Central Processing Unit) came with microfabrication technique. The microfabrication of CPUs increases the surface temperature, which has almost reached to the maximum. As a result, the way of CPU improvement has been changing to increase the number of cores. The CPU with multiple cores is called "multi- (less than 16 cores) / many-core (more than 16 cores) CPU".

Multi- / many-core CPUs have some different points from single-core CPU in its usage. Maximizing calculation speed with the multi- / many-core CPU requires the parallelized software; such as the one which is composed of a lot of small independent tasks, which are distributed to each core. Therefore, it is important to reduce the sequential structure of software as much as possible. The sequential structure slows calculation speed down, known as Amdahl's law.

In computer science field, the main purpose of parallelization is to agree the software before decomposition with the one after exactly. However if the software has large sequential structure, it is difficult to decompose it farther.

In the field of high-performance computing, various mathematical decomposition methods have been developed. It is suitable for using multiple CPUs. The methods such as the QR decomposition by House holder conversion and the LU (Lower Upper) decomposition are known as the ways to be used when using multiple CPUs / cores.

Various methods suitable for embedded multi- / many-core CPU has been studied in recent years. The methods can roughly be divided into 5 levels (see Fig. 1); 1. Control design level, 2. Software code implementation level, 3. Compiler level, 4. OS (Operating System) level, 5. Hardware design level.

In compiler level, effective decomposition methods have been studied; Kasahara et al. proposed the automatic parallelized compiler named "Oscar compiler" (Kasahara 1991), (Umeda 2014). Oscar compiler can quickly deal with changing of a number of cores, but it cannot maximize calculation speed of the engine controllers which aren't designed for multi- /many-core usage because a lot of dependent variables and large sequential structure exist. Another concern is the reliability of the software, which decreases each time when it is decomposed.

Peer review under responsibility of International Federation of Automatic Control.
10.1016/j.ifacol.2016.08.099

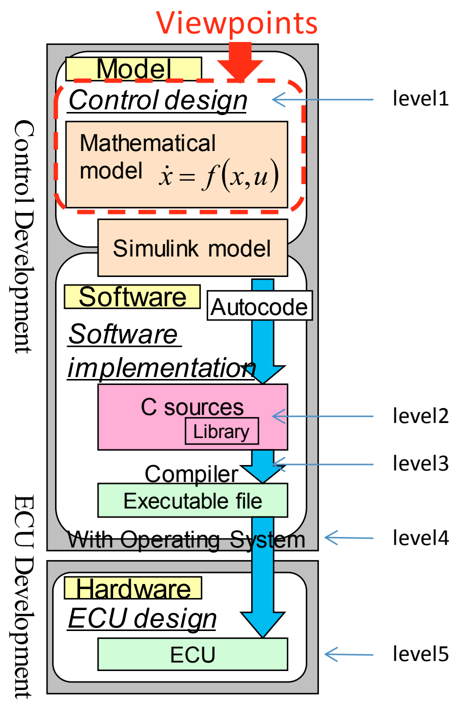

Fig.1 Viewpoints and five levels of parallelization in engine control development process. We propose the parallelization method at Control design phase (level 1).

In the field of control design, several methods to decompose software to the smallest unit have been studied as follows; the adoption of a decomposition method (Samar 2007), fast model predictive control (Wang 2010), parallelized model predictive control (Hara 2014), and parallel design of feedback control systems utilizing dead time for embedded multicore processors (Suzuki 2014). The purpose of the studies is to make the software that does not depend on the number of cores.

The compensation method for numerical calculation is important because the calculation in ECU uses single precision floating point, which isn't precise enough for mathematical calculation; therefore, numerical calculation errors and numerical instability easily occur. However, considering production cost, the calculation using up to single precision floating point is only available with embedded ECU.

In this paper, we propose a decomposition method which can deal with increasing a number of cores to improve calculation speed on production ECU easily. In order to implement software after control design easily, we decompose the controller at control design phase (See Fig.1); the parallelization method is able to decompose software to the minimum unit size. As a result, it enables us to minimize the implementation cost of engine controller, and to use effective control design method on multi- / many-core ECU. Finally, the method can maximize calculation speed to complete calculation within a limited time, assuming that increasing the number of core enhances calculation performance.

We demonstrate the parallelization method with an engine intake system. We suggest a compensation method to resolve the numerical instability of the model. Finally, we check the effect of the parallelization method and evaluate it.

This paper consists of five chapters. We describe a simple engine model of intake flow. In section 3, we derive a parallelized method using state equation. In section 4 simulation results are given and discussed. Finally, this study is summarized in section 5.

2. ENGINE MODEL OF INTAKE PRESSURE

Internal combustion engines have many manipulators including throttle, igniter, intake-VVT (Variable Valve Timing), and exhaust-VVT. Especially more precise intake air flow estimation is important because of large influence of fuel consumption and emission. Therefore air flow mass estimation model becomes more complicated year by year. Recently, many manipulators such as the turbo charger, the super charger and the external EGR (External Gas Recirculation) are embedded to the system, which results in a complicated system structure.

To explore how to manage and decompose the sequential structure, we consider a simple engine model. Fig.1 shows the sketch of an engine.

R is the gas constant of the air, T_a is the atmospheric temperature, p_m is the pressure in the plenum chamber, p_a is the atmospheric pressure, and A_{th} is the throttle valve-opening area. The air mass flow rate of throttle is modelled by

$$\dot{m}_{th} = A_{th} \cdot \frac{p_a}{\sqrt{RT_a}} \phi\left(\frac{p_m}{p_a}\right), \tag{1}$$

where

$$\phi(x) = \begin{cases} \dfrac{1}{\sqrt{2}}, & 0 \le x < 0.5 \\ \sqrt{2x(1-x)}, & 0.5 \le x \le 1. \end{cases} \tag{2}$$

The intake manifold pressure rate is represented as

$$\dot{p}_m = \frac{RT_m}{V_m}\left(\dot{m}_{th} - \dot{m}_{cyl}\right), \tag{3}$$

where V_m is the intake manifold volume, T_m is the gas temperature in the plenum chamber, $\dot{m}_{cyl}$ is the in-cylinder air mass flow rate given by

$$\dot{m}_{cyl} = \eta_v \frac{p_m}{RT_m} \cdot \frac{Ne}{60} \cdot \frac{V_d}{2}, \tag{4}$$

where Ne denotes the engine speed, V_d is the cylinder volume, and η_v is the efficiency of mass flow.