한국컴퓨터정보학회 동계학술대회 논문집 제27권 제1호 (2019. 1)

러너블-태스크 매핑 규칙을 통한 AUTOSAR 기반 차량 시스템의 성능 최적화

민우영[O], 노순현[*], 홍성수[*]
[O*]서울대학교 전기·정보공학부
e-mail:{wymin[O], shnoh[*], sshong[*]}@redwood.snu.ac.kr

Optimizing the Performance of AUTOSAR-based Automotive System via Runnable-to-Task Mapping Rules

Wooyoung Min[O], Soonhyun Noh[*], Seongsoo Hong[*]
[O*]Dept. of Electrical and Computer Engineering, Seoul National University

● 요 약 ●

세계 주요 자동차 회사들은 효율적인 차량용 소프트웨어 개발을 위해 AUTOSAR 표준을 필수로 적용하고 있다. AUTOSAR 기반 소프트웨어의 기능은 러너블(runnable) 단위로 구현되며 이는 태스크에 매핑되어 동작하는데, 러너블-태스크 매핑은 시스템 오버헤드 발생과 러너블의 실제 수행 시점에 크게 영향을 미치므로 시스템 성능 측면에서 매우 중요한 작업이다. 본 논문에서는 자동차의 제어를 보조하는 타겟 응용에 대하여 최적의 성능을 보이는 러너블-태스크 매핑을 찾고자 기존 연구에서 제안된 6개의 매핑 규칙을 적용하며, 기존 규칙의 한계점을 개선한 매핑 규칙을 제안하여 추가로 적용한다. Infineon 사의 AURIX 보드와 ETAS 사의 AUTOSAR 플랫폼 상에 타겟 응용을 구현하여 실험한 결과, 기존 매핑 규칙에 비해 개선된 규칙을 적용하였을 때 종단 간 응답시간이 21.23% 단축되었다.

키워드: 오토사(AUTOSAR), 러너블-태스크 매핑(runnable-to-task mapping), 매핑 규칙(mapping rules)

I. 서론

자동차가 점차 전장화됨에 따라 유럽의 주요 자동차 회사들은 차량용 소프트웨어의 효율적인 개발을 위해 AUTOSAR(AUTomotive Open System ARchitecture)라는 차량용 소프트웨어 아키텍처 표준을 제정하였으며 현재 많은 자동차 회사들이 이를 준수하여 소프트웨어를 개발하고 있다. AUTOSAR에 따르면 응용 소프트웨어는 높은 재사용성과 독립성을 위해 소프트웨어 컴포넌트(software component. 이후 SWC) 단위로 모듈화되어 설계된다. SWC의 기능은 각자가 가지는 러너블(runnable)을 통해 구현되는데, 이는 개발자가 입력한 일련의 코드로써 운영체제의 스케줄링 단위인 태스크에 매핑되어 실행된다[1].

이때, 러너블-태스크 매핑은 AUTOSAR 기반 차량 시스템의 성능을 좌우하는 중요한 작업이다. 몇 개의 태스크를 생성하며 각각에 어떤 러너블을 매핑하는지에 따라 시스템 오버헤드의 양과 러너블의 실제 수행 시점이 달라지며 이는 시스템의 성능에 크게 영향을 미친다. 하지만 러너블-태스크 매핑은 온전히 개발자의 역할이며 AUTOSAR 표준에서는 매우 기본적인 가이드[1]만 제공하므로 개발자가 무수히 많은 매핑의 경우 중 최적의 매핑을 찾기란 매우 어렵다. 이러한 문제를 해결하고자 시스템 성능 측면에서 효율적인 러너블-태스크 매핑을 찾는 연구들이 많이 진행되었다[2, 3, 4, 5, 6, 7, 8]. 기존 연구들이 고려하는 성능 저하 요인으로는 러너블들이 서로 다른 태스크에 매핑될 때 발생하는 문맥 교환 오버헤드, 동기화 오버헤드와 러너블들이 같은 태스크에 매핑될 때 발생하는 수행 지연이 있다.

본 논문에서는 위에서 언급한 성능 저하 요인을 모두 고려하고 AUTOSAR 표준에서 정의하는 러너블의 통신 동작을 기반으로 제시된 [8]의 러너블-태스크 매핑 규칙을 활용하여, 자동차의 안전한 주행을 위해 제어를 보조하는 ADAS(Advanced Driver Assistance System) 응용의 성능을 최적화하는 매핑을 찾고자 한다. [8]의 매핑 규칙은 러너블 간 통신 지연 최소화만을 목표로 하고 있기 때문에 본 논문에서는 타겟 응용의 종단 간 응답시간을 최소화하기 위해 기존 규칙을 발전시키고 발전된 매핑 규칙을 타겟 응용에 적용한다. Infineon 사의 AURIX 보드와 ETAS 사의 AUTOSAR 플랫폼 상에 기존 규칙과 발전된 규칙을 적용한 타겟 응용을 각각 구현하여 종단 간 응답시간(end-to-end response time)을 비교한 결과, 발전된 규칙을 적용할 경우 약 21.23% 단축됨을 확인하였다.

II. 배경지식

이 장에서는 AUTOSAR에 대해 소개한 뒤 [8]의 매핑 규칙의 기반인 러너블의 통신 동작을 설명한다.

1. AUTOSAR 개괄

AUTOSAR 기반 소프트웨어는 설계와 구현 단계로 나누어 개발된다. 먼저 설계 단계에서 개발자는 응용 소프트웨어를 SWC 단위로 모듈화하여 설계한다. SWC의 기능은 내부의 러너블과 이를 작동시키는 이벤트로 설계되며 SWC 간의 통신은 데이터 출입구인 port와 port를 통해 데이터를 주고받는 방식인 interface를 정의함으로써 설계된다.

구현 단계에서는 설계된 사항들이 RTE(Run-Time Environment)로 구현된다. 러너블은 개발자가 내부 코드를 작성할 수 있는 함수로 구현되고 SWC 간의 통신은 RTE API로 생성되어 러너블의 코드 내에서 호출된다. 러너블은 운영체제의 태스크에 매핑된 후 자신을 작동시키는 이벤트가 발생했을 때 태스크 내부에서 호출되어 실행된다. 한 태스크에 여러 러너블이 매핑되면 수행 순서를 설정해야 한다. 태스크는 waiting state의 유무에 따라 기본 태스크와 확장 태스크로 구분되는데 waiting state가 있는 확장 태스크만 수행 도중에 멈출 수 있다.

2. 러너블의 통신 동작

러너블의 통신 동작은 러너블에 부여되는 속성을 통해 정의되며, 이에 따라 RTE API의 구현이 달라진다. 대표적으로 DataSendPoint와 DataReceivePoint는 러너블이 특정 데이터를 송신 또는 수신함을 나타내고 WaitPoint는 러너블이 blocking communica- tion함을 나타낸다. WaitPoint를 가진 러너블은 수행 도중에 최대 timeout만큼의 시간동안 특정 이벤트를 기다릴 수 있다. 러너블이 DataReceivedPoint를 가진다면 가장 최근 수신된 데이터를 읽어오는 RTE API가 생성되고, WaitPoint를 함께 가진다면 데이터가 새롭게 수신될 때 발생하는 이벤트를 기다린 뒤 데이터를 읽어오는 API가 생성된다.

III. 매핑 규칙을 이용한 타겟 응용의 러너블-태스크 매핑

이 장에서는 [8]에서 제시한 6가지 매핑 규칙을 설명한 뒤 이를 본 논문의 타겟 응용에 적용한다.

1. 매핑 규칙 설명

[8]은 러너블 간 통신 지연을 최소화하기 위해 6개의 매핑 규칙을 제시한다. 6개의 규칙은 아래와 같다.

R.1) 데이터 송신 러너블이 하나의 DataSendPoint를 가지며 수신 러너블이 하나의 DataReceivePoint를 가진다면 이 둘은 같은 태스크에 매핑.

R.2) 둘 이상의 러너블들이 같은 데이터를 받기 위해 하나의 DataReceivePoint만을 가진다면, 이들은 같은 태스크에 매핑.

R.3) 둘 이상의 러너블들이 같은 데이터를 받기 위해 하나의 DataReadAccess만을 가진다면, 이들은 같은 기본 태스크에 매핑.

R.4) 같은 주기의 TimingEvent로 인해 작동되는 러너블들이 WaitPoint 없이 모두 동일한 러너블에게 데이터를 보낸다면, 모두 같은 기본 태스크에 매핑.

R.5) canBeInvokedConcurrently 속성이 true이고 WaitPoint가 없는 서버 러너블은 클라이언트 러너블과 같은 태스크에 매핑.

R.6) 서로 다른 WaitPoint를 가진 러너블들은 같은 태스크에 매핑하지 않음.

R.1은 데이터를 송수신하는 러너블 쌍을 한 태스크에 매핑하여 문맥 교환, 동기화 오버헤드를 줄인다. R.2, R.3, R.4는 같은 조건에서 작동하는 러너블을 함께 매핑하고 R.5는 클라이언트와 서버 러너블을 같은 태스크에 매핑하여 문맥 교환 오버헤드를 줄인다. R.6은 한 WaitPoint로 인해 태스크 수행이 멈추면 다른 WaitPoint를 가진 러너블은 기다리는 이벤트가 발생하여도 수행을 재개할 수 없어 생기는 지연을 막는다.

2. 매핑 규칙 적용

본 논문에서 러너블-태스크 매핑을 수행할 타겟 응용은 ADAS 응용으로써 센서로부터 주변 상황과 차량 상태를 입력받아 앞 차와의 안전거리 유지하며 차선을 벗어나지 않게 주행하도록 제어한다. Fig. 1과 같이 12개의 SWC로 구성되며 크게 입력, 추정, 연산, 출력 4단계로 진행된다. 각 SWC는 하나의 러너블을 가지며 러너블의 이름은 SWC의 이름과 같다.

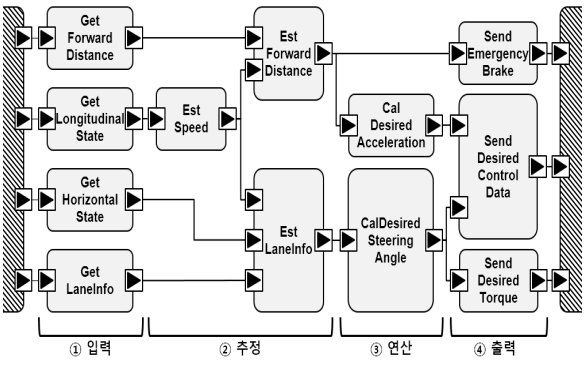

Fig. 1. 타겟 응용의 SWC 설계

입력 단계의 4개 러너블은 앞 차와의 거리, 속도와 가속도, 조향각과 요 레이트, 차선 정보를 입력받으며 센서 주기에 맞춰 수행된다. 추정 단계의 3개 러너블은 10ms마다 제어 신호를 출력하기 위해 앞 차와의 거리, 차량 속도, 차선 정보를 추정한다. 연산 단계의 2개 러너블은 앞 차와 안전거리를 지키고 차선을 벗어나지 않도록 목표 가속도와 조향각 값을 연산한다. 출력 단계의 3개 러너블은 긴급 제동 신호, 목표 가속도와 조향각 값, 목표 핸들 토크 값을 외부로 출력한다. Table 1은 12개 러너블을 명세한 표이다.

Table 1. 러너블 명세

이름	속성	이벤트
GetForwardDistance	DRP&WP, DSP	TE 50ms
GetLongitudinalState	DRP&WP, DSP	TE 20ms
GetHorizontalState	DRP&WP, DSP	TE 10ms
GetLaneInfo	DRP&WP, DSP	TE 100ms
EstSpeed	DRP, DSP	TE 10ms
EstForwardDistance	2 DRP&WP, DSP	TE 10ms
EstLaneInfo	3 DRP&WP, DSP	TE 10ms
CalDesiredAcceleration	DRP, DSP	DR
CalDesiredSteeringAngle	DRP, DSP	DR
SendEmergencyBrake	DRP, DSP	DR
SendDesiredControlData	2 DRP&WP, DSP	TE 10ms
SendDesiredTorque	DRP, DSP	DR

DRP: DataReceivePoint, WP: WaitPoint, DSP: DataSendPoint
TE: TimingEvent, DR: DataReceivedEvent

12개의 러너블에 6개 규칙을 적용하면, Fig. 2와 같이 총 7개의 태스크로 매핑된다. 먼저 Task 2와 Task 6은 R.1에 따라 서로 하나의 데이터를 송수신하는 러너블들을 매핑하였고 Task 5는 R.1과 R.2에 따라 같은 조건으로 작동되는 수신 러너블을 함께 매핑한 후 이들에게 데이터를 송신하는 러너블 또한 함께 매핑하였다. R.6에 의해 WaitPoint를 가진 7개의 러너블은 함께 매핑될 수 없으므로 각각 다른 태스크에 매핑하였다. 하지만 EstForwardDistance와 EstLaneInfo는 동일한 이벤트를 기다리는 WaitPoint를 가지므로 Task 5와 Task 6은 합병할 수 있다. 따라서 기존 매핑 규칙을 적용하면 Fig. 2와 같이 7개의 태스크로 매핑되는 경우와 Task 5, 6을 합병하여 6개의 태스크로 매핑되는 두 가지 경우가 존재한다.

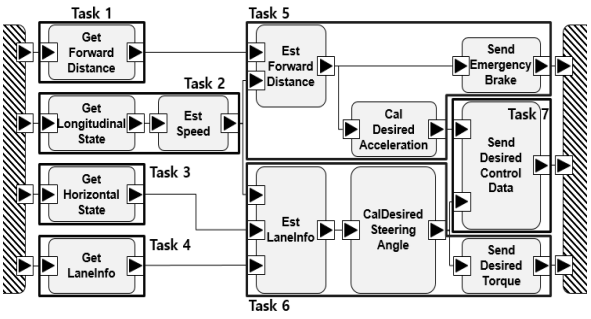

Fig. 2. 기존 규칙에 따른 러너블-태스크 매핑

IV. 기존 매핑 규칙의 개선과 적용

이 장에서는 추가적인 성능 향상을 위해 기존 매핑 규칙을 개선하며 이를 타겟 응용에 적용한다.

1. 기존 매핑 규칙 개선

3장에서 설명한 6개의 매핑 규칙은 통신 지연 감소만을 목표로 하여 제시된 규칙이라는 한계점이 있다. 이 절에서는 종단 간 응답시간을 최소화하기 위해 R.4와 R.6를 확장하고 R.7을 추가한다.

R.4) 같은 주기의 TimingEvent로 인해 작동되는 러너블들은 모두 같은 태스크에 매핑.

R.4는 문맥 교환 오버헤드 때문에 데이터의 송신이 늦어지지 않도록 같은 주기로 작동하는 송신 러너블들을 함께 매핑하는 규칙이다. 이는 송신 러너블이 아니더라도 같은 주기로 실행되는 러너블들이 따로 매핑되면 문맥 교환 오버헤드가 발생하여 종단 간 응답시간이 증가하므로 위와 같이 R.4를 확장한다.

R.6) 서로 다른 WaitPoint를 가진 러너블들은 같은 태스크에 매핑하지 않음. 단, WaitPoint에서 기다리는 이벤트들의 발생 순서가 정해져 있다면 발생 순서에 맞추어 같은 태스크에 매핑 가능

R.6는 한 태스크 내에서 먼저 수행되는 러너블의 WaitPoint에서 이벤트를 기다리고 도중에 다른 러너블의 WaitPoint에서 기다리는 이벤트가 먼저 발생하면, 해당 러너블의 통신이 지연되므로 이를 막기 위한 규칙이다. 이러한 지연은 WaitPoint에서 기다리는 이벤트의 발생 순서와 이들을 가진 러너블의 수행 순서가 동일하다면 일어나지 않으므로 불필요한 문맥 교환 오버헤드를 제거하고자 위와 같이 규칙을 확장한다.

R.7) WaitPoint를 가진 러너블이 같은 태스크 내 존재하는 다른 러너블보다 선행될 필요가 없다면 수행 순서를 마지막으로 배치.

태스크의 수행이 WaitPoint 때문에 멈춰있다면, 이미 자신을 작동시키는 이벤트가 발생하여 수행 가능한 상태인 러너블은 모두 지연된다. 따라서 수행 가능한 러너블이 있음에도 불구하고 태스크가 멈추지 않도록 위와 같이 R.7을 추가한다.

2. 개선된 매핑 규칙 적용

R.4′을 반영하면 10ms로 수행되는 러너블 중 다른 규칙들을 위반하지 않는 EstForwardDistance와 EstLaneInfo가 함께 매핑된다. 따라서 Fig. 2에서 Task 5와 6이 하나로 합병된다. R.6′를 반영하면 EstForwardDistance와 EstLaneInfo의 WaitPoint에서 기다리는 이벤트가 발생한 뒤 연산 단계의 러너블이 수행되고 이후 SendDesiredControlData의 WaitPoint에서 기다리는 이벤트가 발생하므로 Est- ForwardDistance, EstLaneInfo와 SendDesired-ControlData는 함께 매핑될 수 있다. 이때 R.7에 의해 SendDesiredControlData는 수행 순서를 마지막으로 배치해야 한다. 따라서 본 논문에서 개선한 규칙을 적용하면 Fig. 2에서 Task 5, 6, 7이 하나의 태스크로 합병되어 총 5개의 태스크로 매핑된다.

V. 실험 및 검증

이 장에서는 실험을 통해 4가지 경우로 매핑된 타겟 응용을 Infineon 사의 AURIX 보드와 ETAS 사의 AUTOSAR 플랫폼에 구현하여 종단 간 응답시간을 비교한다. 첫 번째 경우는 모든 러너블을 독자적인 태스크에 따로 매핑하여 시스템 오버헤드를 최대화한 경우이고, 두, 세 번째 경우는 [8]의 규칙에 따라 매핑했을 때 7개의 태스크, 6개의 태스크로 매핑된 경우이다. 네 번째 경우가 본 논문에서 개선한 규칙에 따라 매핑한 경우이다. 실험 결과는 Fig. 3과 같다. 본 논문에서 개선한 매핑 규칙을 적용하였을 때 타겟 응용의 종단 간 응답시간이 오버헤드가 가장 큰 경우에 비해 약 42.10% 단축되며, [8]의 규칙을 적용한 경우에 비해 약 21.23% 단축됨을 확인하였다.

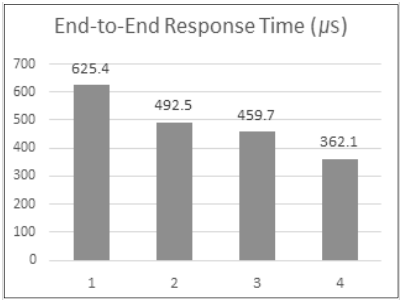

Fig. 3. 실험 결과

VI. 결론

본 논문에서는 AUTOSAR 기반 ADAS 응용에 대해 성능을 최적화하는 러너블-태스크 매핑을 찾고자 [8]의 매핑 규칙을 적용하였으며 추가적인 성능 향상을 위해 이를 확장, 추가하였다. 기존 규칙과 제안된 규칙을 적용하여 러너블-태스크 매핑 작업을 수행했을 때 타겟 응용의 성능을 비교한 결과, 제안된 규칙을 적용했을 때 종단 간 응답시간이 약 21.23% 단축되었다. 향후 연구로는 매핑된 태스크를 멀티코어 상에서 할당하는 연구를 진행할 예정이다.

ACKNOWLEDGEMENT

이 연구는 산업통상자원부 자동차산업핵심기술개발사업 (10079961, “자율주행차 통합제어를 위한 1㎲이내 동기화 성능의 DCU(Domain Control Unit) 제어플랫폼 개발”)의 지원을 받아 수행된 연구결과임.

REFERENCES

[1] AUTOSAR Consortium, AUTOSAR Release 4.3, Specification of RTE Software, 2017.

[2] Wozniak, Ernest, et al. "An optimization approach for the synthesis of AUTOSAR architectures." Emerging Technologies & Factory Automation (ETFA), 2013 IEEE 18th Conference on. IEEE, 2013.

[3] Monot, Aurélien, et al. "Multisource software on multicore automotive ECUs—Combining runnable sequencing with task scheduling." IEEE Transactions on Industrial Electronics 59.10 (2012): 3934-3942.

[4] Faragardi, Hamid Reza, Björn Lisper, and Thomas Nolte. "Towards a communication-efficient mapping of AUTOSAR runnables on multi-cores." Emerging Technologies & Factory Automation (ETFA), 2013 IEEE 18th Conference on. IEEE, 2013.

[5] Zhang, Ming, and Zonghua Gu. "Optimization issues in mapping AUTOSAR components to distributed multithreaded implementations." Rapid System Prototyping (RSP), 2011 22nd IEEE International Symposium on. IEEE, 2011.

[6] Sailer, Andreas, et al. "Optimizing the task allocation step for multi-core processors within AUTOSAR." Applied Electronics (AE), 2013 International Conference on. IEEE, 2013.

[7] Faragardi, Hamid Reza, et al. "A communication- aware solution framework for mapping autosar runnables on multi-core systems." Emerging Tech- nology and Factory Automation (ETFA), 2014 IEEE.

[8] Long, Rongshen, et al. "An approach to optimize intra-ecu communication based on mapping of autosar runnable entities." Embedded Software and Systems, 2009. ICESS'09.