parMERASA

Multi-Core Execution of Parallelised Hard Real-time Applications Supporting Analysability

D6.8 – Recommendation for Parallel Support in Future AUTOSAR Standards to the Standardisation Committees

Nature:	**R - Report**
Dissemination Level:	**PU - Public**
Due date of deliverable:	**31/03/2014**
Actual submission:	**XX/XX/2014**
Responsible beneficiary:	**UAU**
Responsible person:	**Theo Ungerer**

Grant Agreement number:	FP7-287519
Project acronym:	parMERASA
Project title:	Multi-Core Execution of Parallelised Hard Real-Time Applications Supporting Analysability
Project website address:	http://www.parmerasa.eu
Funding Scheme:	STREP SEVENTH FRAMEWORK PROGRAMME THEME ICT – 2011.3.4 Computing Systems
Date of latest version of Annex I against which the assessment will be made:	June 20, 2012
Project start:	October 1, 2011
Duration:	36 month

Project coordinator name, title and organisation: Prof. Dr. Theo Ungerer, University of Augsburg

Tel: + 49-821-598-2350 Fax: + 49-821-598-2359 Email: ungerer@informatik.uni-augsburg.de

Release Approval

name	role	date
Sebastian Kehr	contributor	2014-04-17
Bert Böddeker	contributor	2014-04-17
Theo Ungerer	coordinator	2014-XX-XX

Table of Contents

1 Experiences

One objective of the parMERASA project is to exploit the fine grained embedded automotive application and to achieve maximum reasonable parallelization (WCET of the sequential program divided by the WCET of the parallel program). DENSO Automotive Deutschland GmbH (DNDE) has chosen an engine management system application as case study. Automotive software is typically executed by an OSEK operation system (ISO/TC 22/SC 3, 2009). In many cases this is part of an AUTOSAR (AUTOSAR consortium, 2012) framework. In the following, AUTOSAR terminology is used. AUTOSAR software is structured in Software-Components. The executable entities within the Software-Components are called Runnables. Runnables are mapped to tasks, which are the schedulable entities of the operating system.

We have investigated parallelization on three levels in the case study. The locatable entities for the different levels of granularity are as follows from coarse-grained to fine-grained:

1. Task groups: We have grouped several strongly coupled tasks into groups
2. Runnables within one task group
3. Intra-Runnable instructions

1.1 Distribution of task groups on different clusters

We have allocated complete task groups to guaranteed resource partitions (GRPs). GRPs are clusters of cores that can be treated in isolation during the WCET analysis as much as possible. The task groups in different GRPs should influence the timing of each other as little as possible. Due to the different load of the task groups, a total theoretical speed-up can be calculated as the sum of all WCETs divided by the longest WCET of all task groups.

The theoretical speed-up can only be achieved by non-blocking wait-free communication without overhead. The vast majority in our application has been sender-receiver communication of at most 32 bit values. This can be realized by direct access to shared memory without additional overhead. In case of bigger data structures, we use non-blocking buffers.

Only in case of client-server communication in which the server has an internal state and is accessed from several task groups concurrently, we use critical sections that have potentially blocking effects. But, these are rare (less than 1% of the communication between tasks).

1.2 Runnables within one task group

For the parallelization of a task group, our constraint was to preserve the end-to-end behaviour of the task group: Given the timing and values of all input, the timing and values of the output should be the same before and after parallelization. This implies that all communication between Runnables has to be preserved in the same order.

This can be achieved by time triggered execution of the Runnables. The aforementioned constraints are guaranteed by the fact that a receiving Runnable is always scheduled after the sending Runnable.

To do so, we have exploited the timing analysable properties of the parMERASA hardware and system software which allows the calculation the WCET of each Runnable even if executed concurrently with other Runnables on the parMERASA system. The schedule is constructed based on the WCETs of the Runnables.

If several Runnables don't have mutual execution order constraints, they are potentially scheduled concurrently on different cores.

1.3 Intra-Runnable instructions

During the parallelization of Runnables within one task group, we have often identified a few chained Runnables that require most processing time. In such cases, only very low speed-up could be gained from executing the remaining Runnables on a separate core.

In these cases, the University of Augsburg will investigate more fine grained parallelism contained within single Runnables such as data parallelism or task parallelism of calling different sub-functions. It is planned to annotate parallel constructs, e.g., using openMP directly in the source code.

Our experiences with the parallelization in our use case lead to the following recommendations to enable the parallelization techniques that have been used.

2 Recommendations

2.1 General recommendations

It has been useful to limit the communication paradigms to a very small subset of the possibilities of AUTOSAR:

1. Implicit sender-receiver communication with last-is-best semantics
    a. Support of require-provide-ports
2. Synchronous Client-Server communication

2.2 Communication between task groups

2.2.1 Recommendations for implementation

2.2.1.1 Non-blocking IOC buffers

Description:	IOC should be implemented using non-blocking buffers.
Rationale:	To minimize the timing impact between OS applications, running of different sets of cores (clusters). This allows scalability and predictable on many-core systems.
Use Case:	Many-core system with clustered architecture.
Dependencies:	--

2.3 Runnables within one task group

2.3.1 Recommendations for standardization

2.3.1.1 Time triggered Runnables

In parMERASA, the schedule of the Runnables within a task group is time triggered. This can be implemented by a schedule table that calls one task for each Runnable or it could be more efficiently be implemented with very low jitter by an 'advance' instruction in the RTE code of the task between each two Runnables that waits for the start time of the next Runnable.

Description:	The RTE shall implement 'advance' instructions that wait for a configurable exact start time for a Runnable.
Rationale:	The synchronization overhead needed for the time triggered communication as used in parMERASA depends only on the precision of the synchronicity of the schedule tables on the different cores.
Use Case:	Parallelization by time triggered communication.
Dependencies:	--

2.3.2 Recommendations for implementation

2.3.2.1 Precise synchronous scheduling

Description:	The OS should support schedule tables for each core that run synchronously within few clocks.
Rationale:	The synchronization overhead needed for the time triggered communication as used in parMERASA depends only on the precision of the synchronicity of the schedule tables on the different cores.
Use Case:	Parallelization by time triggered communication.
Dependencies:	--

2.4 Intra Runnable parallelization

The work for Runnable parallelization is still in progress and recommendations will be updated later. Conceptually, the code that has been executed in one Runnable on a single core will be distributed to run on several cores. This can certainly be implemented by creating several new partial Runnables that run on several tasks on different cores, synchronized by explicit AUTOSAR communication.

If the parallelization of Runnables is used frequently, it might be better to provide direct support in AUTOSAR:

- AUTOSAR can (statically) allocate several cores for one Runnable that runs in one task.
- AUTOSAR OS can automatically create several threads from one Runnable and handle the communication between the threads based on the openMP annotations within the Runnable.

3 Summary

The general conclusion is that the methodology of parMERASA can already be implemented in compliance with AUTOSASR 4.1. The main recommendations are for the implementers of AUTOSAR operating systems. To enable WCET analysis, the properties of parMERASA hardware and of the tiny automotive OS should be considered, see (parMERASA, 2013) For practical use of intra-Runnable parallelization, further extensions of AUTOSAR might be desirable.

References

AUTOSAR consortium. (2012, January). *AUTomotive Open System ARchitecture (AUTOSAR).* Retrieved from http://www.autosar.org

ISO/TC 22/SC 3. (2009). *Road vehicles -- Open interface for embedded automotive applications -- Part 3: OSEK/VDX Operating System (OS).*

parMERASA. (2013). *D6.5 – Report on Experiences with Tiny Automotive RTE for Multi-cores to the Standardisation Committees.*