Available online at www.sciencedirect.com

ScienceDirect

IFAC-PapersOnLine 49-25 (2016) 505–510

Tool System and Algorithms for Scheduling of Computations in Integrated Modular Onboard Embedded Systems

Vasily V. Balashov*, Valery A. Kostenko*, Vadim A. Balakhanov*, Sergei A. Tutelian*

* *Department of Computational Mathematics and Cybernetics, Lomonosov Moscow State University, Leninskie Gory, MSU, 1, Bldg. 52, Room 764, Moscow, Russia, (e-mail: {hbd, kost, baldis}@lvk.cs.msu.su, sergei.tutelian@yandex.ru)*

Abstract: Scheduling of computations is an essential step in the process of real-time systems design. In this paper, the scheduling problem is addressed for integrated modular onboard embedded systems (IMOES). This class of systems uses a mix of static and dynamic scheduling. A family of algorithms for workload distribution and schedule construction for IMOES is presented, along with the results of their experimental evaluation on synthetic and real-world data. The algorithms are implemented in a tool system, which is accepted for operation by one of the leading Russian aircraft design companies.

Keywords: Real-Time Multiprocessor Systems; Microprocessor Based Control Systems; Scheduling Algorithms; Design Tools and Application Software.

1. INTRODUCTION

Onboard embedded systems for aircraft and naval purposes are responsible for control of vehicle subsystems and for processing of control tasks such as navigation, collision avoidance, etc. Instead of having federated architecture with dedicated hardware and software for each logical subsystem, the current trend for onboard systems is to run the software of multiple subsystems on a unified platform with standard API and modular hardware, which constitute integrated modular architecture (Wind River / IEEE, 2008). Such integrated modular onboard embedded systems (IMOES) include from several to dozens of modules, each usually containing a multicore CPU. Modules are connected by a network, typically a switched one with support for virtual channels (Schaadt, 2007). Workload for such system consists of a set of periodic tasks with data dependencies. A dependency between tasks corresponds to a message to be transferred between the sender task and the receiver task. In this paper we consider partitioned task sets in which tasks are grouped into subsets (partitions), each of which represents an application.

As IMOES are inherently multiprocessor systems with unified interface for application tasks, a problem of workload scheduling arises, which includes distribution of partitions to CPU cores and construction of partitions execution schedules. IMOES of a modern aircraft runs several hundred tasks with complex data dependencies, so the scheduling problem needs tool support.

In this paper we present a tool system for scheduling of computations in IMOES and describe the scheduling algorithms implemented in this system. Results of tool and algorithms evaluation are also presented, both for synthetic tasks sets and a task set from a real-world IMOES.

This work is partially supported by the Russian Foundation for Basic Research under grant №16-07-01237.

2. STRUCTURE OF THE SCHEDULE

A two-level scheduling scheme is used in IMOES. On the top level, partitions execution is organized via static schedule. For each partition, there is a set of execution windows, i.e. time intervals in which tasks from the partition are executed. We consider systems in which every partition is statically bound to a specific CPU core, thus all execution windows for the partition belong to that core. Partitions binding to CPU cores, as well as the set of execution windows, are to be constructed in advance, prior to the target system startup.

Within an execution window, tasks from the corresponding partition are scheduled dynamically according to their fixed priorities. This constitutes the bottom level of the scheduling scheme. A task which depends on data from another task with the same frequency (from the same or from another partition) can start only after data arrival. This is a synchronous dependency between tasks. A data dependency between tasks with different frequencies does not force the receiver task to wait for input data (asynchronous dependency).

There is a set of constraints on the partitions execution windows defined by the target system (IMOES) specifics. For instance, there can be lower and upper limits on the execution window duration.

3. SCHEDULING PROBLEM

The scheduling problem for a given IMOES and workload (set of periodic tasks grouped into partitions) breaks down into following subproblems:

1) distribute the workload, i.e. bind the partitions to CPU cores;

2) construct the schedule of partitions execution windows.

There are following constraints on partitions binding:

- binding for some partitions can be restricted to a subset of CPU cores, e.g. cores of a specific module;
- total load for a core must not exceed a given limit, which can be defined individually for each core;
- each partition must be bound do a single CPU core.

In course of IMOES evolution, the set of partitions can be extended. So the algorithms for workload distribution must support incremental mode in which previously constructed binding for partitions (all or some of them) cannot be altered.

CPU core load by a partition is calculated as a sum $\sum_i f_i \cdot c_i$, where f_i is the task frequency and c_i is its worst case execution time (WCET). For every task, its WCET is a part of input data for the scheduling problem; for different types of CPU cores used in the target system, WCET for the same task may be different.

We consider workload distribution as an optimization problem with network load as the objective function to be minimized. The network load is calculated as the total size of messages transferred between modules during a single iteration of the schedule, the duration of which is estimated as the least common multiple of the tasks' periods. Messages between tasks running on the same module are transferred through the module's local memory and do not contribute to the network load.

Constraints on the schedule of partitions execution windows are as follows. For each core, the execution windows must not overlap; exactly one partition can be assigned to an execution window; for each core, only partitions bound to this core can be assigned to execution windows for this core; there are lower and upper limits on the execution window duration, common for all cores; the set of execution windows for all cores of the same module must be the same.

The exact set of constraints on workload distribution and on the schedule of partitions execution windows is determined by the specifics of the target IMOES and may vary from system to system. The constraints described above correspond to an IMOES of a modern Russian aircraft and can be considered typical for an onboard computer system.

As the tasks within execution windows are scheduled dynamically, the schedule of windows must guarantee that all of the tasks' executions are performed within deadlines under control of a given dynamic scheduler. Single execution of a periodic task is called a job. If T is the task period (reciprocal of its frequency), i is the number of task iteration (numbering starts from 1), then the deadline interval for the job is $[(i-1)\cdot T; i\cdot T]$.

In terms of jobs, the schedule of windows must guarantee that all jobs of all tasks are executed within corresponding deadline intervals. This can be checked via construction of a job execution sequence taking into account tasks' priorities and data dependencies, and assuming that each job's execution duration (not counting preemption and waiting for input data) equals to task's WCET.

4. OVERVIEW OF EXISTING SOLUTIONS

Since the scheduling problem for IMOES is divided into two sufficiently different subproblems (workload distribution and execution windows schedule construction), we will consider existing solutions for these subproblems separately.

The workload distribution problem for IMOES resembles the multiple container packing problem (MCPP). The latter is the problem of choosing several disjoint subsets of n items to be packed into distinct containers, such that the total value of the selected items is maximized, without exceeding the capacity of each of the containers (Raidl, 1999). In our case partitions correspond to items, CPU cores correspond to containers. Item's volume is the partition's contribution to the CPU core utilization; item's cost is the part of traffic which becomes "internal" for the module when the item is placed in the container (i.e. the partition is assigned to a CPU core). However the problem statement differs from traditional MCPP (with fixed volumes and costs of objects). First, the volume of an item depends on the choice of the container, as a task can have different WCETs for different types of CPU cores. Second, the value of an item depends on the container's contents, i.e. the set of other items in the container.

Due to this difference in the problem statement, existing algorithms for solving of MCPP cannot be used "as is". Following approaches applicable to MCPP were taken as the base for our workload distribution algorithms:

- greedy heuristic search – a common way to quickly find an acceptable solution when the constraints are not too strict; see (Crainic et al, 2012) for example of its application to MCPP;
- branch-and-bound method – looks for the exact optimal solution, working time and scalability greatly depends on the quality of search space pruning; applied to MCPP in (Fukunaga et al, 2007);
- genetic algorithms – chosen for presumably good scalability (opposite to branch-and-bound) and ability to avoid dead ends during the search (opposite to greedy heuristics); applied to MCPP in (Raidl, 1999).

These basic approaches were modified to match the workload distribution problem stated above; see Section 5 for description of the resulting algorithms.

As IMOES, including ARINC 653-based avionics systems, are becoming more widely adopted, there is an increasing amount of research on the techniques for schedule construction for such systems. However in most of the published approaches either the workload is analysed in