Tasking Framework: Parallelization of Computations in Onboard Control Systems

Olaf Maibaum[1], Daniel Lüdtke, Andreas Gerndt

German Aerospace Center - DLR, Simulation and Software Technology

Future unmanned space missions have a great demand on computing resources for the onboard data processing or for control algorithms. Besides the preprocessing of scientific data to reduce the data amount for the downlink, control systems with optical sensors come into play, for example for extraterrestrial navigation and landing systems. Current missions like the Rosetta mission or the landing of the Mars rover Curiosity are based on predefined timed command lists to control the landing. By this and the uncertainty of propulsion and parachute maneuver the amount of different landing targets is reduced to great areas with low risks. But the interesting areas for planetary research have also risky landing areas. So, an autonomous control is needed for the spacecraft to control the trajectory and integrate hazard avoidance algorithms. Such algorithms have a big demand for computing power.

Current onboard systems for the space environment did not provide the needed computing power. Beside this, space systems offer several controller boards on the spacecraft, most of them dedicated to only one subsystem and often twice for cold and hot redundancy. Such designs raise the power consumption and increase budgets like the mass, envelop, and cost. For future missions a new concept for the onboard system is necessary, which allows sharing of computing resources based on predefined configurations for different flight phases and fault scenarios.

With the OBC-NG (On-Board Computer – Next Generation) project DLR started an activity to design onboard computer systems with a combination of space qualified processing node, COTS (Commercial of the shelf) processing nodes and network nodes. The space qualified processing nodes are foreseen for the safety critical processes and the reconfiguration management. COTS processing nodes provide computing power for none safety critical processes. The computation in the different subsystems will share the set of computing resources.

As operating system for OBC-NG an enhancement of RODOS [1] will be used. This enhancement covers mainly the support of multi-core and reconfigurable distributed systems. Core element for this purpose is the Tasking framework. The Tasking framework was designated to simplify the development of the sensor data fusion part in complex attitude control systems. It splits the computations in small pieces, so called tasks, which are scheduled on the availability of the input data.

The first usage of the framework was in the ATON (Autonomous Terrain-based Optical Navigation) project. The project addresses the navigation system for a moon landing scenario. The navigation is based on optical sensors and uses a set of image processing algorithms. The application shows that the Tasking framework is a useful way for the parallelization of computations. The experiences from this project are used for further improvements of the Tasking framework.

The Tasking framework is based on C++ and provides five base classes for the application developer. The developer overloads an abstract virtual method "execute" of the task class with the application-specific computations. This concept is equal to the concept of the RODOS operating system, where a thread is implemented by overloading the thread class. For each task a number of inputs is specified. Each input is associated with a data container, which provides memory space

[1] Deutsches Zentrum für Luft- und Raumfahrt e.V., Simulations- und Softwaretechnik, Lilienthalplatz 7, 38108 Braunschweig. E-Mail: olaf.maibaum@dlr.de

and the synchronization for messages consumed by the task. To drive a task the input is configured with the number of arrivals of the associated message. If the number of arrivals is reached the input is activated. When all inputs of the task are activated, the task is immediately started by the scheduler of the Tasking framework if a free executor, for instance a computing core, is available. If no executor is available the task is queued. Also, an input can marked as final, which starts the task immediately when the input is activated, independent of the state of other inputs of the task. As result a task can produce a new message, which can trigger further tasks. The principle is like a Petri net, where the input is a token and the task is a transition.

To specify timings in the Tasking framework a special message with no data is provided, called event. An event is associated with a clock and sends an empty message with each clock tick. It can be configured as relative or periodic clock. Most of the time the relative clock is used as a time out, where the empty message is associated to a task input with zero arrivals and marked as final. The periodic clock is used to execute a task in a periodic manner.

Tasks can be grouped to sets with a different run time behavior of the inputs. If a task is not associated to a task set, its inputs are reset to zero activations when the execution of the task finishes. The inputs of tasks in a set are reset when all tasks in the task set are executed. Thus, a task can only be executed again if every other task in the set is executed in the previous scheduling cycle.

To support the mapping of tasks in a distributed system, task messages are associated with interfaces to read and write from and to networks and devices. These associations are set up by the configuration manager and are not visible for the application developer. The configuration manager in OBC-NG holds for different hardware configurations of computing resources and mission phases predefined mappings of tasks and resources. Depending of the mappings during configuration phases of the system the tasks and communication infrastructure is set up. An adaptive configuration is possible with this concept but it is not desired for space application, which always should show a deterministic behavior for the operators.

Currently an implementation of the tasking framework on top of POSIX exists, composed by a real time clock interface, signaling mechanism, memory access and the task scheduler. In the near future the RODOS implementation will be updated and ported to the ARM Cortex A9 architecture.

As an additional use case for the application of the Tasking framework the attitude control system (ACS) for the next compact satellite mission in the DLR is selected, beside the ongoing ATON project. The software development for this mission starts in 2014. Main goal of the experiment is to replace all threads in the subsystem by tasks. This will reduce the amount of stack space used by the ACS by a factor of twelve. Also, it simplifies the testing of the software by the lack of endless loops inside the threads.

The presented Tasking framework enables parallel processing of applications for distributed onboard systems. It becomes a core element for the operating system in the OBC-NG hardware, but runs also on top of other operating systems. This allows the mixing of space qualified computing resources with COTS computer platforms for special purposes. By a configuration manager all computing tasks can mapped on the heterogenic distributed system. The scheduling concept of the Tasking framework fits well with the required fixed timing and data driven concept of current used onboard software for space applications. In future work we will evaluate if the concept can also be used to schedule computations on FPGA, DSP, and GPU coprocessors.

[1] *RODOS operating system for Network Centric Core Avionics*. Sergio Montenegro, John Richardson. Proceedings of the DASIA 2009 Conference. Eurospace. Istanbul, 2009.