Distributed Flight Software

V.Vishav[1], A. Lund[2], P. Kenny[2], Z.A. Haj Hammadeh[1], A. Gerndt[1 3] and D. Lüdtke[1]

[1]Institute for Software Technology – German Aerospace Center (DLR)
38108 Braunschweig, Germany

[2]Institute for Software Technology – German Aerospace Center (DLR)
82234 Weßling, Germany

[3] Center for Industrial Mathematics - University of Bremen
28359 Bremen, Germany

As technology advances, space missions are also becoming more and more complex. The number of sensors and component interactions is growing and vast amounts of data are being produced. For this reason, there is an increasing need for data-driven space applications. Complex on-board navigation algorithms with image processing, high-speed data selection and compression applications and identification of potential disasters in Earth observation images using onboard artificial intelligence are some of the applications where enormous amounts of data are processed on-board the spacecraft. To meet the real-time requirements of such applications, high-speed processing of input data is required. This demand for high computational capability is a challenge in space systems. Space missions typically use radiation-hardened space-qualified hardware (e.g. RAD750) which have subpar performance in comparison to standard embedded platforms (e.g. Zynq 7000). This can pose a challenge as it might be difficult to find a single space-qualified processor that can meet such intensive computation requirements.

Some of these types of applications lend themselves well to programming in a data-flow oriented style. In this style, the program is modelled as a directed graph, where data will be processed by operations in the nodes and then forwarded to subsequent nodes in a pipelined manner. Data-driven mechanisms have significant potential for parallelism. The instructions do not have to wait for the preceding tasks to finish but can be executed as soon as the data (operands) are available. This parallelism can be realized with distributed systems. The parallel instructions can be distributed among various computational units and executed simultaneously, allowing faster processing in a multi-processor system.

A software framework to support distributed applications has been realized with the *Tasking Framework* developed by the German Aerospace Centre (DLR). It is a multi-threaded execution platform that uses an *event-driven* approach. With the Tasking Framework, an application can be divided into a directed graph of *tasks* and *channels* which is called a *task map*. A task performs a single process and sends its results to stateful buffers, called channels, which in turn will trigger further tasks. To support distributed execution, the Tasking Framework has been extended for distributed systems. Instead of one computing unit, the task map is now split among different computing units. Each computing unit only executes the task allotted to it and store its results in channels, that may reside on another unit.

An ongoing project at DLR, ScOSA Flight Experiment, implements distributed execution using the Tasking Framework as part of its software stack. ScOSA is a distributed computing architecture with a fault-tolerant, scalable and reconfigurable middleware intended to be used for future space missions. The middleware implements various services as part of its fault detection, isolation and recovery (FDIR) system. Distributed execution works together with the FDIR services to enhance the reliability and fault-tolerance of the system. It also aids in system scalability. The number of computing units can be increased or decreased as required just by changing configuration settings. The heterogeneous nature of the system allows the tasks to communicate on multiple network interfaces (Ethernet and SpaceWire) and platforms (Linux and RTEMS). The project aims to support computationally intensive applications such as real-time image processing and onboard navigation using commercial off-the-shelf (COTS) components.

In this talk we present the class of data-driven applications that are suitable for distributed execution. We give a brief overview of the Tasking Framework and show how we extended it for distributed execution. Finally, we present how some data-driven applications are integrated with distributed execution in ScOSA.