Framework to facilitate development of multithreaded applications in ROS2

While robotics engineers can describe in great detail complex interactions and processes of robots, they struggle to translate these mathematical concepts into software components. This is especially challenging when robotics engineers try to harvest computing potential of multi-core processors. As a consequence, robotic software is often not optimized for concurrent execution, and in many cases, real-time constraints are not met.

As a solution to this problem, we suggest to provide an abstraction that will enable robotics engineers to specify their intentions regarding concurrency. In order to be able to use these intentions and automatically generate software components and set low level properties that enable software to take advantage of concurrent hardware, we will create a set of best practices to instruct robotics engineers on how to develop their software. This way, robotics engineers will mainly worry about defining the components of the robots, whereas, software engineering related decisions are taken by the abstraction provided to them. This sort of the abstraction will be gathered into a framework. This framework will be implemented on the top of ROBOT OPERATING SYSTEM 2 (referred as ROS from here on). ROS is a popular and widely adopted robotics engineering framework. Parameters which will define framework are: low latency, resource utilization (CPU and memory), scheduler information, allocation of memory on heap, process execution priority and threads execution priority. In order to quantify these parameters, we will complement ROS tools that monitor above mentioned information and indicate system bottlenecks.

The main benefit of this work is to enable robotics engineers to develop their robotics applications in a more efficient way, without the need to learn underlying complex concepts of concurrent software engineering. The focus is on assisted usage of multicore processors and concurrency, through automatic and semi-automatic assisted operations. These operations will guide robotics engineers and assist them when possible so they can mainly focus on robotic problems and less on software engineering problems of concurrency. Besides the obvious benefit coming from the concurrent execution of the software, robotics engineers will be spared from learning the configuration of scheduling and shared memory. On top of that, generated concurrency configuration will be robust and portable as a result of using the framework, robotics engineers will not have the overhead of developing the solutions themselves.

In order to achieve these benefits, it will be necessary to first explore what kind of concurrency related operations did ROS already take over from underlying operating system on which it executes, and following the established mindset, recognize perspective for the further extension of ROS considering concurrency. Then, it will be necessary to identify terms and properties that are of importance to robotics engineers through use cases such as inverse kinematics, flight stability, image processing and set of parameters such as execution priorities, deadlines and scheduling that are characteristic for concurrent execution of software from the perspective of software engineering. All research directed towards concurrency, definition of descriptive language and translation of robotics engineers needs will be based on top of node abstraction defined by ROS. Since some concurrency related problems are already solved in ROS, emerging descriptive language will play a role in both optimizing already defined solutions proposed by ROS and implementing new approaches regarding concurrency on top of ROS stack.