Assessment of the Resilience Concept by means of the HiPeRCAR Simulator

Pasquale A. Marra [(1)] (pasquale.marra@aleniaspazio.it)
Daniel Akuatse [(2)] (daniel.akuatse@syderal.ch)
Emily Crudo [(3)] (emily.crudo@galileoavionica.it)
Flavio Fusco [(3)] (flavio.fusco@galileoavionica.it)
Sergio Montenegro [(4)] (sergio.montenegro@first.fraunhofer.de)
Raffaele Vitulli [(5)] (raffaele.vitulli@esa.int)

(1) *Alcatel Alenia Space Italia, SS Padana Superiore, 290 - 20090 Vimodrone (MI), Italy*
(2) *Syderal S.A., Neuenburgstrasse 7 - 3238-Gals (BE), Switzerland*
(3) *Galileo Avionica SpA, Via Montefeltro, 8 - 20156 Milano, Italy*
(4) *FIRST, Kekuléstraße 7 - 12489 Berlin, Germany*
(5) *ESA/ESTEC, Keplerlaan 1, Postbus 299 - 2200 AG Noordwijk, The Netherlands*

INTRODUCTION

Dependability, a combination of availability, reliability and safety, wants to be the characteristic of the HiPeRCAR system. The ESA-funded project HiPeRCAR (High Performance Resilient Computer for Autonomous Robotics) shows how to combine reliability and robustness in an optimal way to get the highest possible dependability using limited resources.
The achieved dependability allows to design a robotic system with suitable computational power for Space.
Keywords: robotics, fault-tolerance, dependability, robustness, resilience, middleware, FDIR.

1. THE HiPeRCAR SYSTEM

HiPeRCAR is a system inspired to the "*MOSREM*" concept by ESA [1], which places radiation-hardened nodes as front-end of a pool of high-performance shear nodes based on industrial processors: the front-end computer provides reliability and continuity to the services; the back-end pool of COTS nodes provides processing power [2].

HiPeRCAR (Fig. 1) combines in a system radiation hardened components with 300-MIPS COTS computers in order to provide the required power to a robotic system in Space.

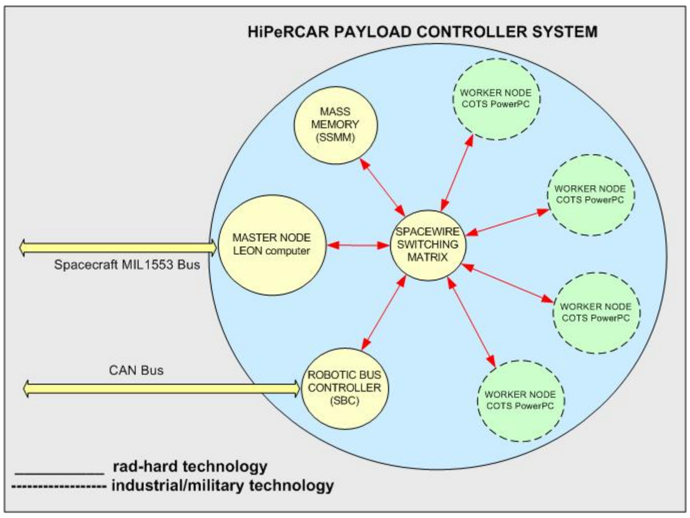

Fig. 1 – HiPeRCAR System

By mixing Space technology and industrial technology, HiPeRCAR is able to offer massive computation-power for trajectory computations, 3-D translations, image elaborations, and so on, and also autonomy and reliability degree to the exploration missions.

A modular design eases the expansion of the system also by introducing additional COTS or redundancy capability. A low-latency high-speed SpaceWire network, a data storing facility and robotic bus controller provide the logistic support to the whole system.

2. THE HiPeRCAR PROJECT

A consortium of European companies lead by AAS-I (Laben) is trying to demonstrate that this goal can be achieved under the ESA contract ITT/1-4607/04/NL/AG.

A first design phase provided the major design features of a possible payload controller system for Space [3]. The second phase of the Project foresees to develop such design. A simulator of a HiPeRCAR system provides the environment for the assessment of the resilience feature. An emulator of a possible HiPeRCAR system will allow soon to measure the performance and to get useful hints to base the design of future robotic missions.

At the current stage, the Project aims at providing an assessment of the resilience strategy with an appropriate apportionment of the tasks over the nodes. This paper shows the results achieved by a simulation of a simplified robotic application.

3. THE RESILIENCE CONCEPT

The realistic scenario of a space mission assumes that failures due to cosmic radiation may hit the Worker nodes; this cause degrades temporarily of definitively the performance of the node in troubles.
HiPeRCAR has to guarantee the absolute continuity of service despite of failures that can hit its weak components. In other words, the system does never interrupt a running service. A global reduction of the performance may be tolerated by it but not loosing of the service. The system must resume autonomously the nominal performance as soon as the critical time is over (i.e. the failure disappears). If the failed node cannot be recovered, the system tries to assign the affected jobs to available nodes or in worst-case carries out by itself those jobs with the possible resources.

4. THE RESILIENCE STRATEGY

The HiPeRCAR system achieves this feature not by complex architectural schemes but through a software technique.
The shutdown and reboot operations of a failed node could last couple of seconds; a time too long to maintain the continuity of the affected service. Therefore, a safe control shall relay on tasks resident on the never interrupted Master node.

The HiPeRCAR Software foresees that reduced versions of the tasks running on the Worker nodes are always active on the Master node. Those basic tasks allow Master to control all the robotic devices by alone, but if the nominal tasks are running on Workers, the results of such advanced services improve movements and autonomy of the robotic system.
If a Worker fails just the contribution of the advanced parts is missing, but the basic control task stays still operable. Thanks to the “Resilience” feature, HiPeRCAR reacts to possible damages of its weakest components by reallocating the faulty tasks in optimal mode. An external observer will not detect discontinuity in the service.

Different design solutions have been tried, mainly addressing the optimal split of the task that has to react instantly to the Worker’s crash. A simulator of the HiPeRCAR system has been set up just to experiment this technique. The out-coming trade-off finds here a description.

5. THE RESILIENCE DESIGN

The design of such a concept goes through the implementation of the following basic tools:

a) Fault Detection and Fault Recovery
   Master node performs the continuous and frequent monitoring of the status of Worker Nodes. The robotic task execution is made by the cooperation of Worker and Master. The presence of the former is not an absolute need.

b) Interconnection network: the Middleware Framework
   A Middleware framework runs onboard all the nodes and carries out in uniform mode all the basic tasks taking care of a heterogeneous environment made of different computer types. Its Communication Manager distributes messages among tasks everywhere located over the network. Middleware permits tasks to smoothly flow from one node to another one. At any time, a task can disappear from one node (e.g. a node crash) and reappear on another node (task distribution) [4].

c) Basic and Nominal Mode for task design
   Basic mode is in charge of the safe operations of the system. Its design is simple, resources economical, in charge to Master Node. While running in basic mode, the system provides a low performance service.
   Nominal mode includes complex and resource-intensive computations provided by a network of tasks distributed in the Worker nodes. The task outputs in Nominal mode are achieved by the contribution of Worker and Master nodes.

The implementation of these basic tools is described in the paper.

Splitting into Basic and Nominal mode for a given robotic function "*F*" is a major challenge of the Project. There are no general rules to split a function into the two complementary parties; this design has to be done on a case-by-case basis. PASTEUR and EUROBOT mission scenarios allow to carry out this exercise.

6. THE HiPeRCAR SIMULATOR

The HiPeCAR generic architecture defined on the Project Phase-1 is simulated on a PC/Linux workstation firstly. Purpose of this simulator is the assessment of the SW architecture, mainly addressing bullets a), b) and c) and the tuning of the algorithms.

The simulator workstation is a PC with a Pentium 4, 3.4 GHz of CPU, 1 GB of RAM and Fedora - Linux Core 5 as Operating System connected via TCP/IP to a PC emulating the current Point Of Control (POC = Ground, Space Station, Spacecraft or Crew MMI) (Fig. 6.1).

A Master, an intelligent Solid-State Mass Memory (SSMM) and two Workers compose the simulated system. Master is in charge of external bus (MIL-1553 and CAN) management. This means that three types of nodes (minimum set) are foreseen:

1. Master that is fixed to be the node number 0
2. SSMM that is fixed to be the node number 1
3. Two Workers that are numbered 2 and 3.

For this reason, three different code images are generated: master image (_m extension); SSMM image (_s extension); worker image (_w extension).
One shell-window is dedicated to each kind of node to run the simulation (Fig. 6.2).

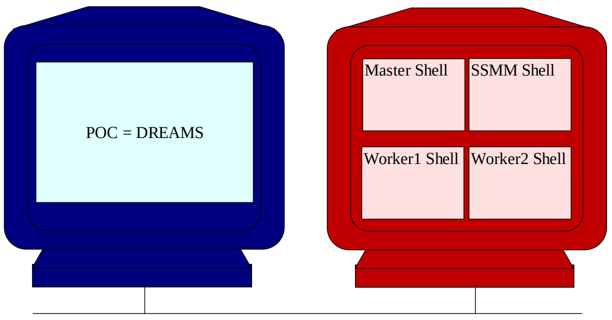

Fig.6.*1*: Simulator Workstation connected to POC PC

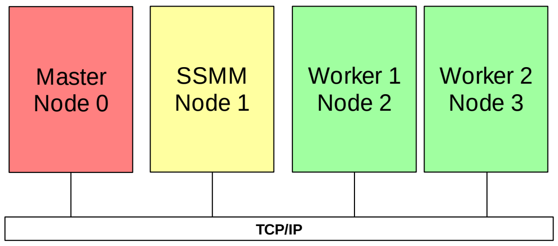

Fig. 6.2: Simulated Nodes connected via TCP/IP

7. TEST SW SIMULATING A REPRESENTATIVE ENVIRONMENT

The HiPeRCAR application layer that runs on top of the Middleware Framework is composed of two parts: Mission Independent, including functions common to all kinds of mission (e.g. Command Handler and Telemetry Manager); Mission Dependent, including the specifying functions of the mission (such as Mission Command Handler and mission Control Tasks, the function that control the foreseen system devices).

The Mission Independent supported functionalities are:

1. External communications, Point of Control handling
2. Commands forwarded to and executed by specialized handlers
3. Telemetry generation
4. Handling of system operative modes
5. Management of Basic and Nominal mode of Control Tasks
6. FDIR-1: cyclic and sporadic checks, centralized event handling, severity thresholds and associated recovery actions
7. FDIR-2: Robustness, resilience and graceful degradation of Control Tasks
8. I/O interface to robotic devices (SCB, science instruments).

In the HiPeRCAR Simulator some extra modules have been added (Fig 7.1) to emulate the external communications with the POC and the Robotic Bus Controller (Servo Control Boards).

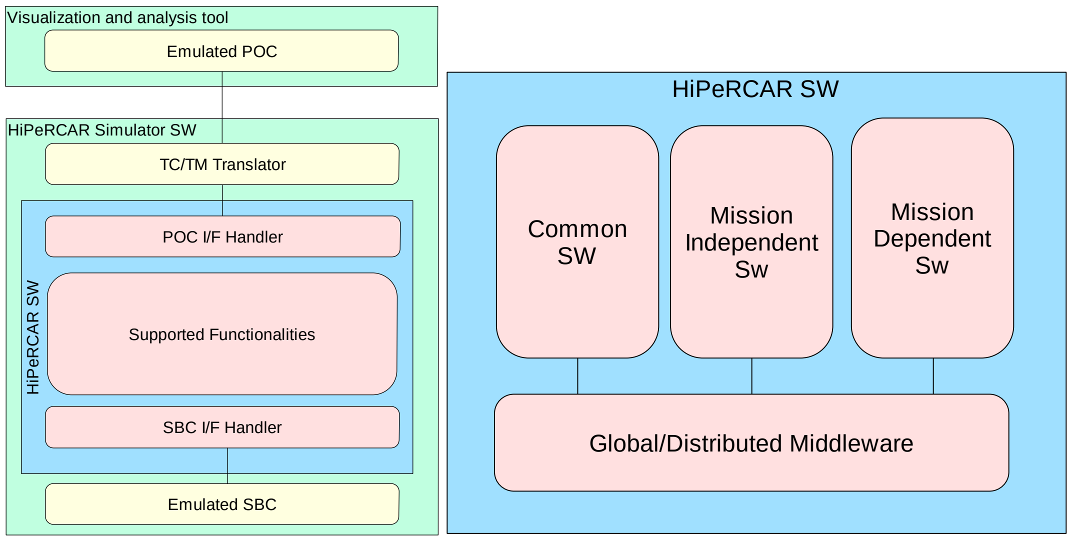

Fig. 7.1: Simulator of HiPeRCAR SW

8. VERIFICATION TEST BED

As HiPeRCAR is a generic platform, in order to demonstrate its features it is necessary to instantiate it for a particular mission. To this purpose, two study cases, inspired to actual ESA missions presently in preparation, are foreseen: PASTEUR and EUROBOT.
For these scenarios the following Mission Dependent functionalities are simulated:
For Eurobot Mission:

Handling of motion state, EUROBOT commands handler telemetry generation, EUROBOT Control Tasks (Control of single arm with EE, Coordinated Control of the arms, Environment reconstruction, Mission Autonomy, Collision Detection, Collision Avoidance, Data Acquisition).

For the Pasteur Mission:

Handling of motion state, PASTEUR commands handler, PASTEUR telemetry generation, PASTEUR Control Tasks (Drill Control, SPDS and Microscope Control, Mission Autonomy, Data Acquisition).

The reference missions are composed of a sequence of tasks. For this reason the task analysis have been performed and a reduced but not simplified subset of the real functions have been identified, so that they may serve as a credible starting point for the specification of the flight software.

After the identification of the reference missions, some test cases have been defined to allow the architecture verification. By injecting of different failures in different execution time on the mission timeline it is possible to activate the reaction of the system:

1. Robustness Test: the mission continues in Basic mode and it is always applied;
2. Resilience Test: the mission continues in Nominal mode recovering the failed worker node or transferring the lost Control tasks into another worker node. It is applied in any case, but not always successfully.
3. Graceful degradation Test: no successful resilience action; therefore, the mission continues in Basic mode.

The paper describes those tests and the achieved results.