On Structuring Holistic Fault Tolerance

Rem Gensh
Newcastle University
Newcastle upon Tyne, UK
r.gensh@newcastle.ac.uk

Alexander Romanovsky
Newcastle University
Newcastle upon Tyne, UK
alexander.romanovsky@newcastle.ac.uk

Alex Yakovlev
Newcastle University
Newcastle upon Tyne, UK
alex.yakovlev@newcastle.ac.uk

Abstract

Computer systems are developed taking into account that they should be easily maintained in the future. It is one of the main requirements for the sound architectural design. The existing approaches to introducing fault tolerance rely on recursive system structuring out of functional components – this typically results in non-optimal fault tolerance. The paper proposes a vision of structuring complex many-core systems by introducing a special component supporting system-wide fault tolerance coordination. The component acts as a central module making decisions about fault tolerance strategies to be implemented by individual system components depending on the performance and energy requirements specified as system operating modes.

Categories and Subject Descriptors D.2.4 [*Software/Program Verification*]: Reliability

Keywords Many-core systems, system structuring, system layering, energy efficiency, performance, error recovery

1. Introduction

All traditional ways of ensuring system fault tolerance are associated with a component structuring. The most prominent forms are nested atomic transactions, nested recovery blocks and nested exception handling. The techniques like these help the developers to focus on providing fault tolerance only at one layer/component, either by tolerating the faults (errors) or, when this is not possible, by passing the responsibility to a higher level. This view is best captured by the architectural style called Idealised Fault Tolerant Component [1]. These recursive approaches are widely used because they allow the developers to drastically reduce the complexity of providing system fault tolerance.

The main problem with these solutions is that they do not support a system-wide view on ensuring fault tolerance, causing system overdesign and resulting in systems that are not optimal with respect to system performance or energy consumption. These issues can be addressed by implementing a vision of holistic fault tolerance (HFT), which introduces a new type of cross-system module capturing system-wide fault tolerance. The module complements the local fault tolerance of the individual components and ensures that the system-wide recovery strategies are implemented. HFT not only simplifies the analysis and implementation of critical system parts, but also facilitates system reuse and elaboration of system operating modes. The paper reports on our design of a medium-scale case study that implements HFT and outlines the challenges to be addressed in engineering HFT while designing many-core systems.

MODULARITY'16, March 14–17, 2016, Málaga, Spain
© 2016 ACM. 978-1-4503-3995-7/16/03...$15.00
http://dx.doi.org/10.1145/2889443.2889458

2. Cross-Layer Fault Tolerance

The notion of cross-layer fault tolerance (CLFT) introduced in earlier work (e.g. [2]) supports the idea that fault tolerance should be cooperatively designed for all system layers. If a system is developed according to this approach, then in case of an error, the entire system stack will collaborate to perform error recovery, since all layers have access to system state-related information. When the standard layered approach is applied, optimality (e.g. with respect to system performance and energy consumption) will be achievable only at separate layers, without guarantying the optimal operation of the entire system. This was the main motivation for introducing CLFT.

In our work, we rely on Cross-Layer Reliability Visioning Study [2], which motivates a new approach to system reliability based on a cross-layer system design. The growing energy demand and the forecasted reliability deterioration of modern hardware ground the importance of the cross-layer approach to reliable system design. CLFT can be applied to cope with these problems [3]. Only with a cross-layer approach, we can make a decision whether the error at the lower layer is critical for the higher layers and apply the most suitable error recovery scenario according to the systems state. Thus, CLFT allows us to decrease error recovery overheads and reduce energy consumption, be-

cause in case of error the optimal fault tolerance strategy will be chosen. CLFT will be very useful for performance- and energy-critical systems.

The cross-layer design is widespread in the area of the wireless sensor networks (WSNs). Wireless sensors are small devices, which measure temperature, humidity, air quality, etc. These sensors are networked to each other and transmit their data to the main server. Very often the battery is the only power supply for them. Therefore, reliability, performance and energy consumption are the most important requirements for these systems. The optimal operation of the WSN can be guaranteed only when the layers of the system stack are considered together. The layered approach does not provide an easy way to share an important information about the system state among the system layers, this makes it impossible to achieve the optimal system operation. The advantages of the cross-layer design for WSNs are discussed in [4]. In spite of the success in applying the cross-layer approach, there is no work on systematic engineering of CLFT in the domain of WSNs. The developers design systems without any support for reuse, often producing similar solutions independently.

The TCP/IP stack illustrates how the cross-layer approach can be applied to ensure fault tolerance and improve performance. All four layers of the TCP/IP stack participate in error detection and error recovery in a concerted fashion. The Link, Internet and Transport layers use CRC checksum to detect and reject corrupted data packets. In addition, the TCP provides data retransmission in case when the sender did not receive the acknowledgement of successful TCP packet delivery. There is a scenario demonstrating cross-layer error detection and error recovery. Let us consider the corruption of the Ethernet frame. The data transmission error will be detected at the Link layer (incorrect CRC checksum) and at the Transport layer (TCP sender did not receive acknowledgement). Error recovery will be performed by the same two layers. The Link layer will reject the corrupted packet and the Transport layer will retransmit the packet by timeout. If the developer chooses UDP as the transport protocol then error detection and error recovery are to be implemented at the application layer, since, unlike TCP, UDP does not provide reliability features. In this case, the errors, which are not detected at the lower layers will be detected at the application layer. Depending on the error type, the error recovery could be performed either only by application layer or in a cross-layer manner by combined efforts of several layers. These examples illustrate how CLFT is applied to ensure efficient and reliable data transmission over the network.

Paper [2] does not discuss any specific techniques or methods, which could be applied for developing cross-layer reliability. In our previous work [3], we introduced the idea of CLFT in the context of many-core systems and briefly outlined some engineering techniques to be used in developing CLFT in this domain. During the recent work that followed we realised that the idea of CLFT needs to be completely rethought to support a system-wide view on introducing fault tolerance in the form of *holistic fault tolerance* (HFT) that is not linked to system layering, but applicable to any systems structured out of components.

3. Number Plate Recognition

Many-core architectures are expected to be widely applied in the next generation of computer systems. The main challenges for many-core systems are to ensure high performance, low energy consumption, efficient resource utilisation and high reliability, thus, it is necessary to find and analyse possible trade-offs between these parameters.

A car number plate recognition (CNPR) application is being developed to gain experience in engineering HFT for many-core systems. HFT is employed to ensure optimal system operation, since energy efficiency is a crucial factor for high-performance applications. The HFT component acts as a central unit analysing the detected errors and choosing the error recovery strategies implemented by the combined efforts of several components. Three possible operating modes are available: reliability, performance and energy efficiency. They differ in the CPU usage, resource utilization, number of possible error recovery attempts, quality of input data and required quality of output. Several implementations of the optical character recognition (OCR) algorithm provide redundancy to ensure reliable CNPR.

The architecture consists of six functional components. The CNPR component plays the role of an intermediary between the user and the system: it receives the image file and passes it to the initial image-processing component. At this stage, the image is being processed by several sequential actions: resolution and contrast adjusting, localization of the number plate, elimination of rotation and perspective skew. At the next step, a cutout of the number plate is passed to the character segmentation component, where the image is divided into several pieces with single character in each. The OCR component receives separated images and launches concurrent OCR. Several recognition algorithms are executed concurrently if the reliability mode is specified. Finally, the recognized symbols are checked by the result checker component. If the result is correct, the recognized string is returned to the user. The probability of the correct recognition is computed as well. The HFT component coordinates error recovery involving these components. Fig.1 shows a bird's-eye view of the design.

These are our fault assumptions. The application should be able to recover the following errors: a CPU error (operation timeout or exception during calculations), insufficient quality of service (i.e. probability of the correct recognition is low), impossibility to detect the number plate at the initial image processing step, a recognition algorithm error

and a result checking error. There are two types of the HFT strategies applied to recover the system.

The following strategies are independent of the chosen operating mode. If the error rate of certain CPU core is much larger than the error rate of other cores, then the HFT component will reconfigure the system and hide the faulty core for predefined time. If the OCR component signals the insufficient quality of provided images, the HFT component will relaunch initial image processing without reducing the image resolution at the initial image-processing step. If the rate of this error becomes very high, the HFT component will completely cancel any reduction of image resolution by the initial image-processing component.

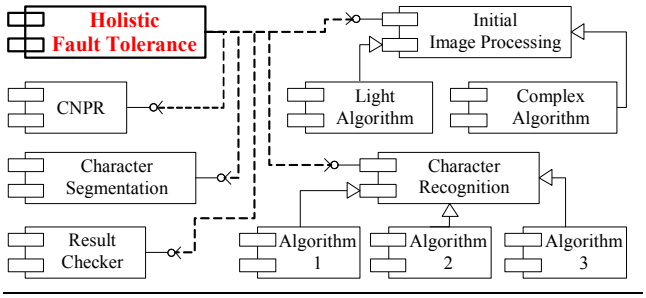

Figure 1. The HFT component in the system.

Fault tolerance strategies, which depend on the operating mode, are described as a decision table stored in the HFT component. For each combination of operating mode and error, there is a certain error recovery scenario. For example, when the CPU error is detected in the performance mode during the work of a recognition algorithm, HFT decides to skip the error, if other recognition algorithm already returned a correct result. In other modes, the operation is re-executed on another CPU core. The next example is a result checking error in the reliability mode. Voting is applied to check if results of the recognition algorithms are different. If the three algorithms return different results, a recovery attempt should be launched, starting from the initial image-processing step. Table 1 summarises possible error recovery strategies depending on the operating mode (the grey cells represent HFT).

The introduction of the HFT component in this medium-scale application has clearly simplified the design of the fault tolerance strategies and eased modification and improvement of system fault tolerance during system development, since most part of the changes related to fault tolerance were made inside a single component. The case study demonstrates the ability of HFT to improve the system performance and energy efficiency. This is made possible because the HFT component has a global view of the system and chooses the optimal fault tolerance strategy. However, this approach breaks the abstraction principles and requires a dedicated engineering support. In the next section we consider possible ways of assisting developers with designing HFT clearly, systematically and efficiently.

Table 1. Error recovery strategies for the operating modes.

	Reliability mode	Performance mode	Energy saving mode
CPU error	Re-execute on another core	Skip the error	Re-execute on another core
Insufficient quality	Run initial image processing again		Run another recognition algorithm. If all algorithms returned an incorrect result, go back to the initial image processing
Recognition algorithm error	Skip error if other algorithms already returned correct result		
Result mismatches the national format	If the attempt is allowed, start from initial image processing	Correct the number at the result checker component	

4. Our Vision

There is interesting evidence indicating that the TCP/IP design that motivates our work suffers from certain oversights because arguably there were some mistakes made in designing CLFT [5]. In certain cases, error recovery can be inefficient due to the lack of collaboration between the Link and Transport layers, as the TCP considers timeouts to be a consequence of high network load and slows data retransmission accordingly. In wireless networks the packet can be lost due to environmental conditions and user mobility, so the absence of information about an error at the Transport layer can cause data transmission delays. This means that, for HFT to be efficient, practical and reusable, its engineering needs to be properly supported.

We believe that in the near future the developers of complex many-core systems will use a dedicated module to capture the essence of holistic fault tolerance which consists of system-wide error detection and error recovery that involve various system components. They will create such systems together with a dedicated HFT module that will define fault tolerance policies which crosscut system functionalities.

Various approaches can be used to support the HFT crosscutting design, including aspects, dedicated reference architectures, extended programming languages, programming libraries, patterns and styles.

Model-based development will play a crucial role in applying HFT in the many-core systems domain. This will allow analysis/verification at the earlier architectural phases and support direct traceability of designs to critical system requirements, such as safety, reliability, energy efficiency and system performance, which can only be met by applying fault tolerance. Note that other non-functional requirements, such as confidentiality and execution cost, are related to fault tolerance and the use of redundancy, too, so they can also be met by designing HFT.

Modelling of HFT can be supported by the mechanisms available for enabling architectural viewpoints (e.g. [6, 7]), because HFT is in effect an architectural view on system-

wide fault tolerance. The links between these two concepts will need to be explored more deeply in the future.

It is clearly not practical to develop HFT that deals with all possible errors in the system. Its complexity would be overwhelming. The practical question is to decide which errors to handle locally in the components and which to handle holistically. The mechanisms to be developed should support both types of fault tolerance and their smooth integration and interplay.

The modelling techniques for HFT should support zooming into system models to help with exploring various HFT solutions and choosing which system components to involve in HFT (again, it is not practical to link HFT with all system components). One promising technique to ensure HFT scalability is Order Graphs [8], which support a rigorous selection of the level of detail at which models are reasoned about. In terms of energy consumption the efficient way is to focus only on the components that consume substantial amounts of energy (see our previous work on modelling many-core system fidelity [9]).

The HFT module will describe a set of system-wide fault tolerance policies along the lines of a well-established area [1]. This will require a domain-specific notation that can represent types of faults and errors and their combinations, the corresponding error recovery and fault handling techniques, resources required, sets of system components involved and propagation chains. The application of such a DSL will help HFT developers to make their work less error prone and more reusable.

To explicitly control energy consumption in many-core systems, HFT should be able to use a number of knobs for energy control in different system components. This includes switching off/on cores, excluding cores at the OS level (e.g. using Linux governors), using DVFS, etc. In addition, various application-specific solutions can be employed: using less precise computation or reducing the quality of the outputs. These need to be supported as well.

We believe that associating modes with various levels of performance and energy consumption is a useful and practical way of structuring HFT in complex many-core applications. As has been shown [7], fault tolerance can be linked with the system modes in a natural way. To this end, engineering mechanisms for mode design and analysis that are associated with HFT will need to be developed.

HFT always relies on a centralised design in which the entire HFT is defined in one module but there might be various ways of implementing the HFT module. The centralised component is the first option but this can create a single point of failure if the component crashes. This can be solved by component replication and checkpointing. Another option is to use aspect-oriented programming. The third possible option is to attach the HFT component to every system component, so that each component is controlled by its own controller.

As part of the HFT design we strongly recommend producing a table that would compile all the faults that have to be dealt with at the system level and the corresponding error detection and error recovery strategies that can involve several system components. It is important to include in this table the erroneous situations when two or more faults can happen in parallel.

There are several other structuring approaches that are different from the traditional caller-callee layered architectures but they cannot fully support our vision of HFT. Brooks' approach [10] to structuring complex control system using levels of competence is an interesting solution but it supports neither component (in this case, level) coordination nor activities that require dynamic decisions which groups of components to involve in fault tolerance. The Integrated Modular Avionics architecture is not developed for modular coordination either. These two approaches are not intended for dealing with situations when several system components need to be involved in system-wide fault tolerance in an orchestrated fashion.

References

[1] T. Anderson, P. A. Lee, Fault tolerance, principles and practice, Prentice/Hall International. 1981.

[2] A. DeHon, N. Carter, H. Quinn, Editors, Final Report for CCC Cross-Layer Reliability Visioning Study, CCC, 2011.

[3] R. Gensh, A. Romanovsky, A. Yakovlev, Engineering Cross-Layer Fault Tolerance in Many-Core Systems, in Proc. *Software Engineering for Resilient Systems*, LNCS-9274, 2015.

[4] Y. Wang, H. Wu, F. Lin, N. Tzeng, Cross-Layer Protocol Design and Optimization for Delay/Fault-Tolerant Mobile Sensor Networks, *IEEE Journal on Selected Areas in Communications* 26(5) 809-819, 2008.

[5] R. Ludwig, Eliminating Inefficient Cross-Layer Interactions in Wireless Networking, PhD Thesis, RWTH Aahen, Germany, 2000.

[6] R. Hilliard, I. Malavolta, H. Muccini, P. Pelliccione, On the Composition and Reuse of Viewpoints across Architecture Frameworks, in *Proc. WICSA/ECSA 2012*, Helsinki, Finland, ACM.

[7] I. Lopatkin, A. Iliasov, A. Romanovsky, Rigorous Development of Dependable Systems using Fault Tolerance Views, in *Proc. ISSRE'11*, Hiroshima, Japan. 2011. IEEE CS.

[8] A. Rafiev, F. Xia, A. Iliasov, R. Gensh, A. Aalsaud, A. Romanovsky, A. Yakovlev, Order Graphs and Cross-layer Parametric Significance-driven Modelling, in *Proc. ACSD'15*, Brussels, Belgium, 2015. IEEE CS.

[9] A. Rafiev, F. Xia, A. Iliasov, R. Gensh, A. Aalsaud, A. Romanovsky, A. Yakovlev, Power-proportional modelling fidelity, in *Proc. 1st Workshop on Model-Implementation Fidelity*, DATE 2015. Grenoble, France. 2015.

[10] R. Brooks, A robust layered control system for a mobile robot, *IEEE Journal of Robotics and Automation* 2(1), 1986.