Secure deployment in trusted many-core architectures

Maria Méndez Real, Guy Gogniat, Adel Baganne
Laboratoire Lab-STICC
Université de Bretagne-Sud
Lorient, FRANCE
Name.surname@univ-ubs.fr

***Abstract*—Secure execution of parallel applications in many-core architecture represents a major concern. Current solutions have not sufficiently addressed this issue which may reduce the adoption of this technology in a near future. In this paper the problem of secure application deployment in trusted many-core architectures is discussed. The Trusted Computed Base (TCB) required to build an efficient and safe solution is first analyzed. Then different execution scenarios and their associated thread model are explored. Finally some important properties of a secure deployment in terms of scheduling, resource allocation and control mechanisms are highlighted. The aim of this paper is to discuss the main requirements to guarantee a trusted execution in many-core architectures.**

Keywords—Many-core architectures; secure deployment; scheduling; thread model

I. Introduction

Secure handling of personal and private data within many-core architectures is a major issue. Economic and social challenges are numerous since many-core architectures are expected to be massively deployed both in cloud computing infrastructures and in performance and resources constrained embedded systems. In addition to performance, it is mandatory to address the question of security when building many-core architectures in order to ensure the adoption of these technologies by end users.

In order to propose an efficient and secure solution it is necessary to develop hardware many-core architectures integrating heterogeneous processing resources. These architectures will thus combine different resources, some CPUs dedicated to the processing of clear data and some cryptoprocessors dedicated to the processing of protected data. It is also important to develop new techniques for dynamic applications deployment, scheduling and resource management algorithms taking into account security needs of many-core architectures.

This paper aims to define a Trusted Computing Base (TCB), to explore the associated threat model and to analyze the properties and feasibility of a secure dynamic applications deployment in many-core architectures. To address this issue, the rest of the paper is organized as follows. Section II first discusses related work. Then Section III presents problem statement and analyzes the properties and feasibility of a secure deployment in many-core architectures. Finally Section IV draws some conclusions.

II. Related work

Many works have focused on the dynamic allocation problem for many-core architectures [1-3]. However, there is no work that directly addresses the problem of secure dynamic applications deployment on many-core architectures. Among previous efforts, Masti et al. [4] provides an interesting contribution. Authors describe a trusted scheduling architecture for embedded systems that ensures the intended execution schedule even when the system is facing some attacks. They assume that hardware components can be trusted and thus they do not consider physical attacks. Their main idea is to reduce as much as possible the TCB by implementing some of its functions in hardware whenever it is possible. This way the kernel faces a lower risk of being compromised. In contrast, this hardware implementation reduces the system flexibility and scalability. In fact, the hardware scheduler, for example, supposes the use of a fixed set of applications with peripheral needs known in advance. Furthermore, additional hardware components dedicated for security imply important changes in the architecture. Finally they do not provide some scheduling algorithms.

Masti et al. [5] explore the security properties and opportunities enabled by many-core systems. Authors modify and extend the existing Intel Single-chip Cloud Computer many-core platform with security properties such as runtime isolation using a small security kernel that constitutes the software TCB. In this case, the functions of the kernel are limited to the assignment of applications to their respective cores and memory. In this way the kernel faces a lower risk of being compromised and integrity and confidentiality attacks are prevented thanks to application isolation.

Moreover, in order to limit the impact of a compromised subset of applications, they introduce interesting security properties like application context awareness. The idea is that applications become aware of which other applications share their resources and have access to their memory space. Thus applications should be able to detect if other applications are compromised and even protect themselves. Finally, they address the problem of application isolation from a possible compromised security kernel. These security properties are interesting but seem complex to implement.

The aim of this work is to explore the threat model associ-

ated to the definition of a TCB and to investigate the properties and feasibility of secure dynamic applications deployment in many-core architectures. This work discusses some foundation in order to build a trusted many-core architecture integrating some cryptoprocessors.

III. Secure dynamic deployment

In this section generic many-core architecture and operating system are first introduced. Then the TCB definition is discussed and its associated threat model is analyzed. Different application scenarios are also proposed to illustrate potential execution schemes on a many-core architecture. Finally required properties and feasibility of a trusted dynamic application deployment are explored.

A. Problem statement

Our work relies on a generic many-core architecture based on logical clusters connected through a Network-on-Chip (bottom side of Figure 1). A logical cluster is composed of one to several physical clusters. Each physical cluster contains some cores, memory-caches and peripherals connected by a local interconnect. Additionally, a core has a memory management unit (MMU) that checks, for each memory access, its type (read, write, execute) and verifies whether or not the entity initiating such an access has the appropriate privileges. Finally, in order to propose an efficient and secure solution, this architecture is extended with a cryptoprocessor in each physical cluster.

An application is defined as a set of tasks that can be efficiently executed in parallel. Each application has a required security level and might need cryptographic computations. Different application execution scenarios can be envisioned.

First, there could be several applications requiring a high security level but with no need of cryptographic computation.

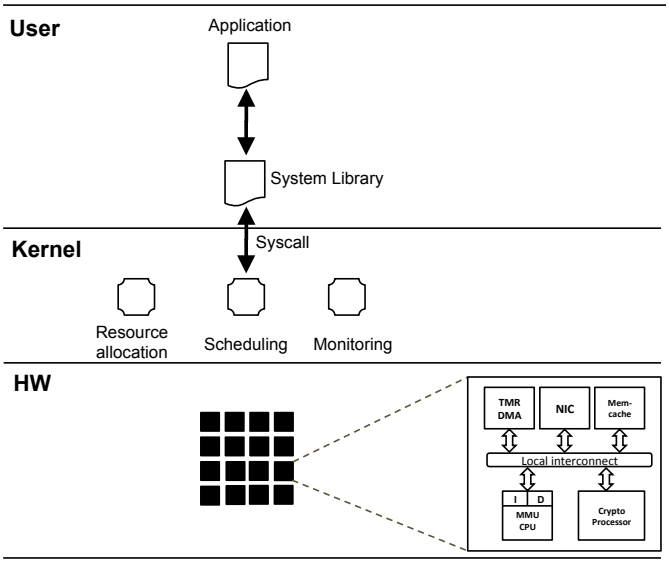

Fig. 1. System layer overview

In this case, logical and physical isolation are necessary. In fact, since applications are sensitive, they need logical isolation to be executed in an isolated and protected environment. This can be achieved through the use of MMU and Operating System (OS). MMU enforces access control policies and checks each memory access privileges whereas the OS is responsible for allocating a separate memory region for each application. Physical isolation is also necessary in order to prevent any leakage of information between applications, which can be achieved through resource exclusivity (cores, cryptoprocessors) for critical applications. Physical isolation can be performed by dedicated hardware firewalls that filter each transfer within the system.

Another scenario could be several applications requiring a high security level and cryptographic computation. In this case it is also necessary to dynamically and securely configure the cryptoprocessors. The OS should be in charge of this new responsibility.

Finally, if applications do not require a high security level (they are not critical), they can share all the resources.

Applications co-exist and share platform resources through an OS that performs the three main functions:

Scheduling: The OS, taking into account the priority of applications, decides the order in which applications are executed.

Resource allocation: The OS is responsible for allocating resources, typically CPU, cryptoprocessors and memory. It allocates resources to each task the scheduler activates and to tasks requesting for more resources during their execution.

Monitoring: The monitoring function provides some useful metrics for allocation such as core utilization rate or the number of active tasks per physical cluster.

The kernel is the first to be executed after the system boot. The kernel first decides when and over which core(s) an application will be executed (according to its scheduling policy). Then the kernel allocates resources, mostly cores and memory and allows the application execution. Finally during its execution, an application can request more resources from the kernel.

Since the TCB runs with the highest privilege level, it is important to keep it as small as possible in order to limit the exposed attack surface. This preliminary study considers that the TCB encompasses the entire OS security kernel. It is also assumed that hardware platform is trusted and attackers can only tamper with the system using logical attacks. Thus physical threats are not considered.

As hardware is trusted and TCB is encompassed by the OS security kernel, four types of attacks can be considered: confidentiality, integrity, denial of services and information leakage attacks, and two main threat scenarios can be highlighted:

There is a single malicious application: A compromised application could launch denial of service attacks by requesting a large number of resources (such as memory, cores, and peripherals) with the objective of saturating the system. However, in this case confidentiality and integrity attacks are impossible since MMU checks privileges of each memory

access. Moreover, a compromised application could try to extract some information by reading peripheral output registers after other application has used them.

There are several compromised applications: compromised applications could address a large number of requests to a given peripheral in order to launch denial of services attacks.

B. Defining Secure Dynamic DeploymentProblem statement

A secure dynamic deployment relies on ensuring isolation and protection of applications requiring a high security level. Thus the expected execution of critical applications is guaranteed even in a compromised environment. Important properties of a secure dynamic deployment in terms of scheduling, resources allocation and control mechanisms are the following ones.

Scheduling and resources allocation: The scheduler policy needs to be preemptive in order to stop after an interruption caused by a possible malicious application.

Moreover, the allocation algorithm needs to take into account the required security level to decide in which cluster each application will be mapped. In fact, in the case where applications with different security levels are executed in parallel two situations can be considered. First, if they do not communicate with each other, then a solution to prevent interference with the critical application would be to place them in physical clusters far away from each other. On the contrary, if they communicate with each other they need to be placed in nearby clusters for performance, and a mechanism to protect these communications needs to be established.

Furthermore, in order to physically isolate sensitive applications, they can be placed in a logic cluster which size is dynamically modified according to the number and needs of sensitive applications.

Control and security mechanisms: In order to prevent logical attacks, control mechanisms are required. First, to avoid denial of services attacks, a maximum CPU and cryptoprocessor utilization time and load must be defined. A maximal memory space can also be allocated. Thus, an attacker cannot preempt system resource required by another application preventing it from running.

Moreover, if communications between applications are known in advance, a monitor could check each communication and detect if an application is sharing resources with any other core contrary to what has been established.

Any violation of these control measures would raise an interruption and the concerned application would be aborted. After an interruption, or when a sensitive application releases memory, the latter needs to be cleared in order to avoid other applications from reading it (information leakage).

Finally, if a single application uses several cryptoprocessors an efficient and secure key exchange of a shared key is necessary.

These secure deployment properties are summarized in Table 1.

Services	**Secure deployment properties**		
	Main functions	**Potential attack**	**Information required to prevent from potential attacks**
Scheduling and resources allocation	Scheduling	DoS[a]	Scheduling policy
	Tasks placement	C & I[b]	Security level required Task resources needs Tasks communication Global system situation
	Dynamic resource allocation	C & I	Security level required Task resources needs Tasks communication Global system situation
Control	Control of maximal resources utilization	DoS	Maximum CPU and crypto processor utilization time. Maximum memory allocation. Maximum number of threads
Security	Context awareness	C & I	Tasks communication Applications resources needs
	Reset resources after an IT or after sensitive application execution	C & I	Security level required
	Protect communications between applications requiring different security levels	C & I	Security level required Tasks communication
	Securely sharing crypto processor key	C & I	Security level required Tasks communication

[a]Denial of services attack
[b]Confidentiality and integrity attacks

TABLE I
SECURE DEPLOYMENT PROPERTIES

IV. CONCLUSIONS

In this paper we define and present a first analysis of the problem of securely deploying applications in trusted many-core architectures. We define the TCB and different application execution scenarios. Then we explore the associated threat model and analyze the properties and needs of a secure deployment in many-cores. A reduced TCB composed only by security and performance critical functions will be considered in further work. Security aware deployment algorithms will be also analyzed.

ACKNOWLEDGMENT

This work was realized in the frame of the TSUNAMY project number ANR-13-INSE-0002-02 supported by the French Agence Nationale de la Recherche [6].

REFERENCES

[1] M. Fattah et al., Cona: dynamic application mapping for congestion reduction in many-core systems", IEEE 30th International Conference on Computer Design,2012, pp. 364-370.
[2] S. Widermann et al., Game theoretic analysis of decentralized core allocation schemes on many-core systems", Design Automation and Test in Europe Conference and Exhibition, 2013, pp. 1498-1503.
[3] R. Das et al., Application-to-core mapping policies to reduce memory system interference in multi-core systems", Proceedings of the 21st international conference on Parallel architectures and compilation techniques, 2013, pp. 455-456.
[4] R. Masti et al., Enabling trusted scheduling in embedded systems",Proceedings of the 28th Annual Computer Security Applications Conference, 2012, pp. 61-70.
[5] R. Masti et al., Isolated execution in many-core architectures, Network and Distributed System Security Simposium, 2014.
[6] The TSUNAMY project www.tsunamy.fr