ALMOS many-core operating system extension with new secure-enable mechanisms for dynamic creation of secure zones

Maria Méndez Real, Vincent Migliore, Vianney Lapotre, Guy Gogniat
Univ. Bretagne-Sud, UMR 6285, Lab-STICC, F-56100 Lorient, France
maria.mendez@univ-ubs.fr

***Abstract*—Many-core architectures are becoming a major execution platform in order to face the increasing number of applications to be executed in parallel. Such an approach is very attractive in order to offer users with high performance. However it introduces some key challenges in terms of security as some malicious applications may compromise the whole system. A defense-in-depth approach relying on hardware and software mechanisms is thus mandatory to increase the level of protection. This work focuses on the Operating System (OS) level and proposes a set of operating system services able to dynamically create physical isolated secure zones for sensitive applications in many-core platforms. These services are integrated into the ALMOS OS deployed in the TSAR many-core architecture, and evaluated in terms of security level and induced performance overhead.**

I. Introduction

Many-core architectures correspond to a main evolution of computing systems due to their high processing power. Many applications can be executed in parallel which provides users with a very efficient technology. However, it introduces key challenges in terms of security as malicious applications may compromise other applications or the whole system by spying, retrieving or injecting infected data. Moreover, with the large number of resources and applications running in parallel, the complexity of handling resources in many-core architectures has widely increased. Current mono and multiprocessor solutions are no longer sufficient to answer the security requirements of many-core architectures and need to be extended or redesigned in order to take into account the scale of many-core architectures where the number of applications running in parallel is much wider. This work focuses on the OS level and proposes a set of operating system services able to dynamically create physical isolated secure zones for sensitive applications in many-core platforms. These new OS services are integrated into the ALMOS OS [1] deployed in the TSAR many-core architecture [2], and evaluated in terms of security level and induced performance overhead. The main contributions of this work are:

- a set of new secure-enable OS mechanisms for the ALMOS OS able to dynamically create secure zones in the TSAR many-core architecture in order to logically and physically isolate sensitive applications avoiding Denial of Services (DoS) and cache Side Channel Attacks (SCA),
- the validation of these services and the evaluation of the performance overhead induced in the ALMOS OS and global performance through the simulation of the ALMOS-TSAR environment.

The rest of this paper is organized as follows. Section II presents the related work. Section III introduces the TSAR architecture, ALMOS OS, applications and the threat model associated to our system. Section IV first explains the principle of physical application isolation and details the extension of ALMOS in order to integrate the new secure-enable services. In Section V, the experimental protocol is presented and experimental results are discussed. Finally, Section VI draws some conclusions.

II. Related work

There is little work that addresses the problem of secure dynamic deployment of parallel applications on many-core architectures. In this section, related work in multi-core and many-core architectures are presented.

Platform bi-partition and MMU/MPU. ARM TrustZone [3] provides hardware support for the creation of Trusted Execution Environments (TEEs) and therefore the isolation of applications within the same processor. This feature creates 2 virtual processors and 2 Memory Management Units (MMU) allowing a ”secure world” to live alongside a ”non-secure world”. However, at any time instant, only a single domain in the system can be secured. Therefore, the isolation is only achieved at the processor level. The use of MMU and Memory Protection Unit (MPU) allows the secure partition of shared memory mediating memory access in order to avoid confidentiality and Integrity attacks (C&I). Nevertheless, in multicore and many-core architectures, applications running on different processors share resources such as the communication infrastructure (NoC, buses) and memory. Bi-partitioning, nor MMU, MPU are not enough anymore. Indeed, applications running on different processors are not protected from each other since sharing the communication infrastructure leads to possible leakage of information attacks such as SCA.

Data protection in NoC based multicore architectures. In [4], authors present a ”NoC MPU” for shared memory NoC based multicore architectures in order to isolate the shared memory partitions. In the architecture proposed, each NoC node is encompassed by whether an initiator or a target device.

NoC MPUs are located at the initiator side network interfaces. The partition access rights tables are configured by the OS. This solution avoids C&I attacks, however, DoS and SCA remain possible.

Fully virtualization. In fully virtualization, the creation of multiple domains, each one being a virtual machine running its own OS instance, is controlled by a trusted software layer. Each virtual machine is isolated avoiding any interaction with other virtual machines. However, within a virtual machine, applications are still sharing resources and are not protected against other applications.

Security capabilities in modern multicore and many-core architectures. In [5], authors propose "illegal access probe" and "denial of services probe". These solutions tackle C&I and DoS attacks. In [6], authors explore the security opportunities enabled by existing many-core systems. They propose the extension of the Intel Single-chip Cloud Computer many-core platform with security properties such as isolation running a trusted agent (Trusting Computing Base (TCB) element) on every core of a many-core platform. Moreover, a new security property called application awareness in order to allow each application to protect itself from a compromised kernel is introduced. This new property is interesting, nevertheless, no actual implementation has been proposed. Modern architectures such as the POWER architecture [7] provide hardware-enforced access mechanisms for memory regions through storage protection keys necessary to access a protected segment of memory countering C&I attacks. While these solutions avoid C&I, they do not counter attacks such as DoS nor leakage of information and more precisely cache SCA [8][9].

Modern OS features that can be used for security purposes. Finally, in some modern OSes such as AIX [10], some features as exclusiveness of computing resources to guarantee resources for an important work are proposed. This kind of features could be integrated with memory and communication protection mechanisms for isolation of applications.

Compared to these efforts, this work focuses on the OS level and proposes a set of OS services able to dynamically create physical isolated secure zones, similar to TEEs but running in parallel in many-core platforms in order to avoid DoS and cache SCA. The aim of this work is to extend the ALMOS OS services in order to integrate these new services, and to evaluate their performance overhead.

III. Problem statement

A. TSAR architecture

TSAR [2] is a many-core architecture made up to 1024 cores based on physical clusters connected through a 2D-Mesh network-on-chip. Each physical cluster contains four cores, a memory-cache per core (Level 1 cache), one physical memory bank (Level 2 cache), and peripherals connected by a local interconnect. This architecture is also extended with a crypto-processor in each physical cluster [11] (Figure 1). The TSAR architecture is non uniform memory access. Each cluster memory bank is actually implemented as a memory

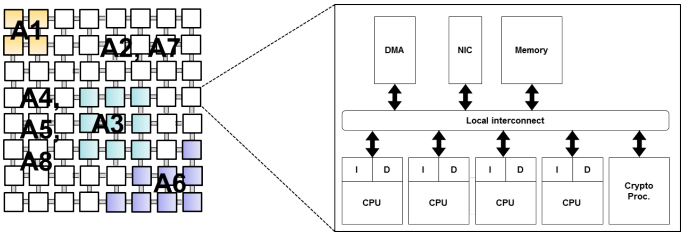

Fig. 1. The TSAR architecture

cache. The latter is not a classical level 2 cache. The physical address space is statically split into fixed size segments (one per physical cluster), and each memory cache is responsible for one segment. This implies that memory in each cluster is actually physically stuck together. This peculiarity facilitates the isolation of memory.

B. ALMOS OS

ALMOS (Advanced Locality Management Operating System) [1] relies primarily on monitoring mechanisms, a scheduling policy, and memory and computing resources allocation services. ALMOS main goal is to enforce the locality of memory accesses made by threads of parallel applications. ALMOS has a distributed approach in order to locally manage the requests (memory allocation, thread and fork mapping) of each physical cluster and satisfy them in the same cluster whenever is possible. In this work, one CPU executes the ALMOS OS in each physical cluster.

C. Application model

Applications are modeled as acyclic directed task graphs that might be intended to be isolated. Each task is characterized by an amount of work in terms of number of processor or cryptographic instructions, an amount of data needed to be allocated, a number of memory accesses and data dependencies.

D. TCB and threat model

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 study considers that the TCB encompasses the entire OS security kernel. It is also assumed that the hardware platform is trusted and attackers can only tamper with the system using logical attacks. The increasing number of applications running in parallel sharing resources introduces some key challenges in terms of security. Three types of attacks can then be considered: C&I, DoS and leakage of information attacks, more precisely communication and cache SCA. In this work communication SCA are not considered. Since C&I attacks are avoided by logical isolation (MMU), two main threat scenarios can be highlighted:

DoS attacks: A compromised application could launch DoS attacks by requesting a large number of resources (memory or computing) e.g. creating a large number of threads, with the

objective of saturating the system or decreasing the performance of other applications needing these resources.

Leakage of information: A compromised application sharing resources with other applications could launch leakage of information attacks and more specifically cache SCA [8][9]. In fact, the cache forms a shared resource that all applications compete for. While the data stored in the cache is protected by the MMU, the memory accesses patterns of the applications using the cache are not fully protected and can be analyzed by malicious applications in order to extract information.

IV. New ALMOS OS services able to dynamically create secure zones physically isolating sensitive applications

Applications are vulnerable when sharing resources. One solution to counter DoS and cache SCA as well, is to execute sensitive applications logically and physically isolated avoiding any resource sharing.

A. Logical and physical isolation:

In order to guarantee a secure deployment of applications on the TSAR many-core architecture avoiding DoS and SCA, we propose to extend ALMOS OS services in order to perform a physically isolated execution of sensitive applications. To illustrate this point, let's consider 8 applications running concurrently on the same platform (Fig. 1), and sharing resources. The MMU guarantees that no application can have access to a physical memory space without the corresponding rights, avoiding C&I attacks. However, DoS and cache SCA are still possible since applications can share processors and memory resources [8][9]. Therefore we propose the dynamic creation of secure zones in order to isolate sensitive applications. Secure zones are composed of a number of contiguous clusters, in which all the resources are dedicated to one single sensitive application. In consequence, this application will not share its resources and thus SCA and DoS of these resources are avoided. This sensitive application will thus be isolated in a logical and physical way (A1, A3 and A6 in Fig. 1). Once the application execution in a secure zone is finished, the secure zone is dissolved and resources are released and reset in order to avoid any leakage of information. However, since some of the resources within a secure zone might not be used or not be used during the entire application execution time, an under-utilization of resources and thus a global performance overhead are expected.

In this work it is considered that each physically isolated application entirely fits into the clusters dedicated to its secure zone. Otherwise, the remotely accesses outside the secure zone need to be protected. There have been some efforts on the protection of communication through network-on-chip [12]. Such solutions can be adapted and integrated in our system in order to protect communications outside the secure zones (this point will be considered in future work).

Algorithm 1 Creating a Secure zone

```
1: Input: The architecture A and the number of physical clusters needed by the relative
   application NbR
2: Output: The set of physical clusters forming a secure zone SZ[NbR]
3: i=0
4: while i < NbR OR all available physical cluster ∈ SZ crossed do
5:     for all available physical clusters C_xy ∈ A do
6:         L[i] = C_xy
7:         if i < NbR then
8:             for all Clusters ∈ SZ do
9:                 while i < NbR do
10:                    for all its 4 physical contiguous neighbor clusters do
11:                        if i < NbR and this neighbor cluster is idle then
12:                            i ++
13:                            SZ[i] ← this neighbor cluster
14:                        end if
15:                    end for
16:                end while
17:            end for
18:        end if
19:        if i < NbR then
20:            empty the SZ and restart from another available physical cluster
21:        end if
22:    end for
23: end while
```

B. Extension of the ALMOS OS in order to integrate the new security-enable services

Several services of the ALMOS OS have been extended or created to take into account the security issues previously explained.

1) Extension of monitoring services: In order to create secure zones, monitoring mechanisms providing information about the global state of the physical platform and available resources in terms of physical memory pages, active tasks and processor utilization rate are implemented. These monitoring services have been extended in order to take into account the crypto-processors that have been integrated in each physical cluster. Moreover, periodic monitoring mechanisms updates have been extended in order to update clusters that have been tagged as dedicated to a secure zone. Monitoring services are consulted each time a resource allocation decision needs to be taken (e.g. new application mapping, new secure zone creation, new task mapping and memory allocation).

2) New secure zone creation service: A new service dealing with the dynamic creation of secure zones has been implemented (Algorithm 1). We consider that the maximum parallelism of each application is known, meaning the maximum number of needed resources (computation and physical memory pages) in order to achieve the application maximum performance. The list-scheduling algorithm searches for contiguous idle clusters (line 10) in order to build a secure zone in which all the resources will be dedicated to the sensitive application during all its execution time. Once the application is finished, the resources are released. For performance purposes, the algorithm tries to create a square-shaped secure zone in order to leverage data locality. If this is not possible, the algorithm settles for contiguous clusters. If there are no physical idle contiguous clusters enough, the algorithm fails on creating a secure zone, and it will try again at the next scheduling tick.

3) Extension of resources allocation services: Resources allocation algorithms (e.g. new application mapping, new se-

cure zone creation, new task mapping and memory allocation) have been adapted to take into account the new parameter of physical clusters meaning that they are dedicated to an isolated application. Indeed, in order to take a memory allocation or mapping decision, the ALMOS OS consults each physical cluster monitoring information until finding the needed resource. A request from a secure zone can only be satisfied inside the secure zone. In the same way, the resource allocation algorithm can satisfy a resource allocation request for a non-isolated application only on non dedicated tagged clusters. As a consequence, the exploration zone is reduced in both cases and bound by the size of the entire platform. Thus, with secure zones, the time spent on resource allocation services is expected to decrease.

V. Results

In order to efficiently address the early evaluation of the new OS services an evaluation tool targeting the ALMOS OS and TSAR many-core architecture has been developed [13]. This tool is used in order to compare the original ALMOS services with the security enhanced ALMOS services on the TSAR many-core architecture. This latter is composed of 4×4 clusters, each containing 4 cores (i.e. 64 cores in total). The main objective in these experimentations is to evaluate the impact of the creation of secure zones in terms of performance overhead induced on the isolated application as well as on the entire execution, both taking into account the overhead on the OS services. In this work, we use synthetic applications whose task graphs are representative of parallelism oriented applications. Each application encompasses 12 tasks running in parallel (12 cores are necessary to achieve one application maximum parallelism). Each task corresponds to 2k core instructions. The applications are duplicated in order to stress the platform. Finally, for these experimentations, the simulation time is up to 17 seconds for 150 000 000 processor cycles simulated.

Performance overhead results:
In this subsection, different scenarios are evaluated to measure the performance overhead induced by the extension of the ALMOS OS services, in each running application and in terms of global system performance.

1. Comparison of the complexity of each ALMOS services for two scenarios: first one single application running on the platform with no security mechanism, and secondly the same application being physically isolated. This, according to the size of one isolated application secure zone (Figure 2): The new application mapping when there is no security mechanisms, secure zone creation, tasks mapping and memory allocation complexities depend on the isolation scenario. The time spent on these three services is lower than the original ALMOS services with any security mechanisms. The secure zone creation is less complex than ALMOS application mapping since there is no load on the architecture. As a consequence, this is the best case of the secure zone creation algorithm. On the other hand, when there are several possible clusters each one just as good as the others, as in this case,

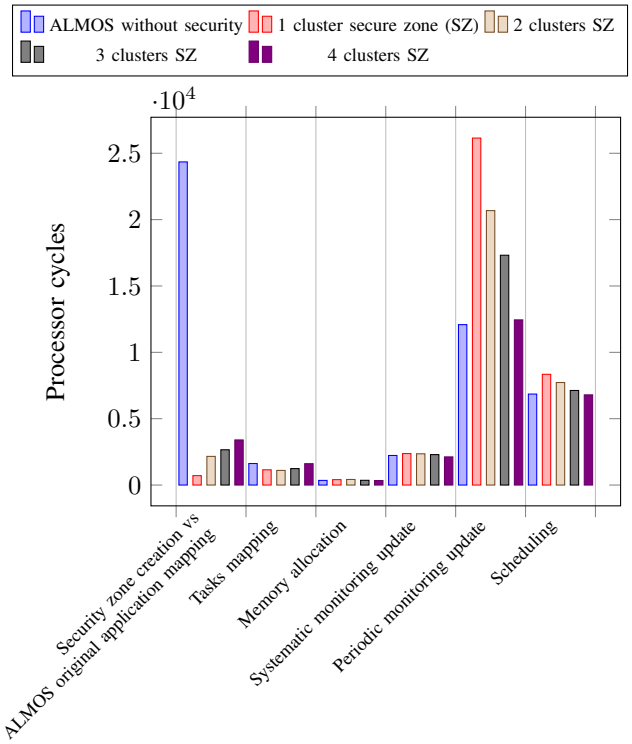

Fig. 2. Comparison of the complexity of ALMOS services with and without security, according to the size of one isolated application secure zone

the original ALMOS new application mapping compares all the candidates in terms of idle CPUs, physical memory pages, and resources utilization rate. This scenario is one of the worst scenarios for this algorithm. The systematic monitoring update complexity is the same for both isolation scenarios since it only depends on the number of tasks running on the platform. Finally, the scheduling and the periodic monitoring update are two OS periodically services. This means that the time spent on these services only depends on the size of the architecture and on the total execution time. Therefore, when the secure zone size is one physical cluster, the time spent on these two OS services has significantly increased since with a single cluster secure zone instead of four, the application execution time significantly increases as well.

2. Comparison of the time spent on the ALMOS services for both scenarios, when no application is physically isolated, and when one single application is physically isolated. The comparison is made according to the load on the architecture (Figure 3): Experimental results show that the overall time spent on the OS services is always lower with physical isolation mechanisms, and can be reduced down to 45%, 8%, 13.5% and 12.7% when there are respectively 1, 5, 10 and 20 applications running on the platform (knowing that for 20 applications, the computing resources utilization rate is up to 56% on average).

3. Evaluation and comparison of the total execution time of non-isolated applications for both isolation scenarios, when no application is physically isolated and when one single

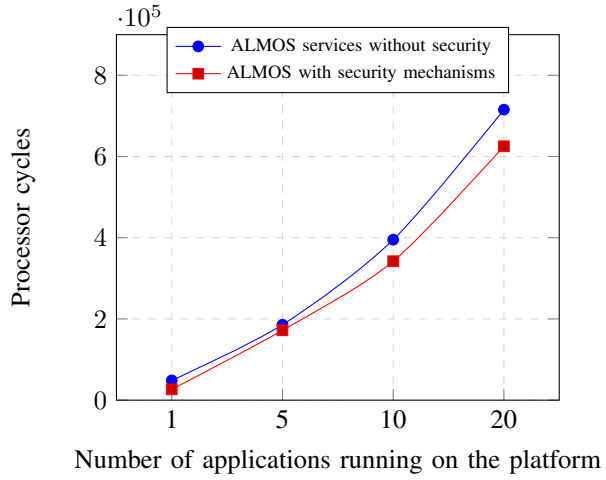

Fig. 3. Comparison of the complexity of ALMOS services with and without security mechanisms according to the load of the platform when a single application is physically isolated in a 4 clusters secure zone

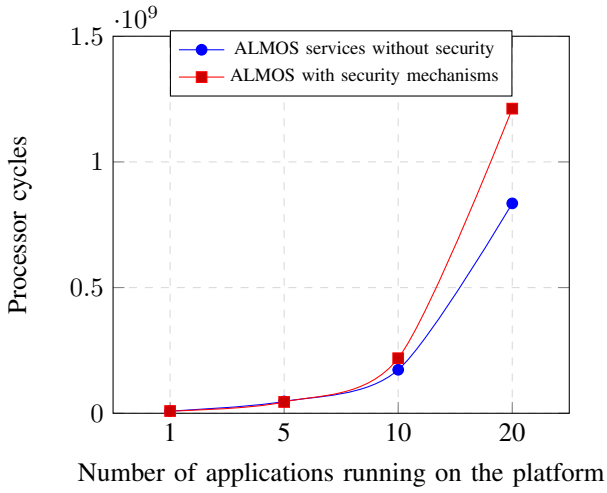

Fig. 4. Comparison of the execution time of non-isolated applications with and without security mechanisms according to the number of applications running on load of the platform when a single application is physically isolated in a 4 clusters secure zone

application is physically isolated, according to the load on the architecture (Figure 4): Experimental results show that the overhead on the total execution time of non-isolated applications increases with the load of the architecture, up to 26% and 45% when there are respectively 10 and 20 applications running on the platform. This because non-isolated applications are deprived of resources dedicated to the secure zone, and the more applications are running on the platform, the more applications share the same number of resources.

Discussion and future work:
The sensitive applications being executed physically isolated in a secure zone, any resource sharing is possible. Thus, DoS or cache SCA are avoided within a secure zone. On the other hand, experimental results show that these new services induce a performance overhead on the execution time of non-isolated applications that increases with the number of applications running on the platform (up to 45% for 20 applications which represents 280 tasks). However, the induced performance overhead is negligible when the architecture load is lower than 26%. Finally, integration of the physical isolation mechanisms on the ALMOS code induces a code size overhead of 9.5% compared to the original ALMOS code. Notice that in this work the size of secure zones is fixed during the entire application execution time. This static approach might cause fragmentation on the platform. A solution in order to avoid fragmentation and to reduce the under-utilization of resources is the dynamic allocation and releasing of resources within secure zones. This new approach will be considered in future work.

VI. Conclusion

In this paper we propose the extension of the ALMOS OS with new secure-enable services able to physically execute sensitive applications through dynamic creation of secure zones for the TSAR many-core architecture. This solution has been validated and evaluated through several experiments implemented within our ALMOS-TSAR evaluation tool. Experimental results show that physical isolation mechanisms avoid DoS and cache SCA. While an overhead on the total execution time of non-isolated applications, depending on the load of the architecture, is highlighted, the complexity and thus the time spent on OS services is reduced. In future work dynamic allocation and releasing of resources within a secure zone will be explored in order to reduce the performance overhead resulting from isolation mechanisms.

References

[1] "ALMOS," www-soc.lip6.fr/trac/almos.
[2] "TSAR," www-soc.lip6.fr/trac/tsar/.
[3] "ARM TrustZone," www.arm.com/products/processors/technologies/trustzone.
[4] G. Kornaros, I. Christoforakis, O. Tomoutzoglo, D. Bakoyiannis, K. Vazakopoulou, M. Grammatikaki, and P. Ao, "Hardware support for cost-effective system-level protection in multi-core socs," in *Proc. of Digital System Design (DSD)*, 2015.
[5] L. Fiorin, G. Palermo, and S. Co, "A security monitoring service for nocs," in *Proc. of 6th International conference on Hardware/Software codesign and system synthesis (CODES+ISSS)*, 2008, pp. 197–202.
[6] R. Masti, D. Rai, C. Marforio, and S. Capkun, "Isolated execution in many-core architectures," in *Proc. of Network and Distributed System Security Simposium (NDSS)*, 2014.
[7] "The POWER architecture," www.ibm.co/1Kj2y2I.
[8] Y. Wang and G. Suh, "Efficient timing channel protection for on-chip networks," in *Proc. of the 2012 IEEE/ACM Sixth International Symposium on Networks-on-Chip (NOCS)*, 2012, pp. 142–151.
[9] J. Demme and S. Sethumadhavan, "Side-channel vulnerability metrics: Svf vs. csv," in *Proc. of 11th Annual Workshop on Duplicating, Deconstructing and Debunking (WDDD)*, 2014.
[10] "AIX Operating System," www-01.ibm.com/support/knowledgecenter/ssw_aix_61/com.htm.
[11] C. Lopez, M. Méndez Real, L. Bossuet, G. Gogniat, V. Fischer, and A. Baganne, "Trusted computing using enhanced manycore architectures with cryptoprocessors," in *Proc. of IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*, 2014.
[12] J. Sepulveda, G. Gogniat, C. Pedraza, R. Pires, W. Chau, and M. Strum, "Hierarchical noc-based security for mp-soc dynamic protection," in *Proc. of Circuits and Systems (LASCAS)*, 2012.
[13] M. Méndez Real, V. Migliore, V. Lapotre, and G. Gogniat, "Evaluation of security services within the almos operating system for the tsar many-core architecture," Submitted at International symposium on circuits and systems (ISCAS), 2016.