Université Pierre et Marie Curie
Laboratoire d'Informatique de Paris 6

Exécution sécurisée de plusieurs machines virtuelles sur une plateforme Manycore

Soutenance de thèse

06 juillet 2017

Clément DÉVIGNE

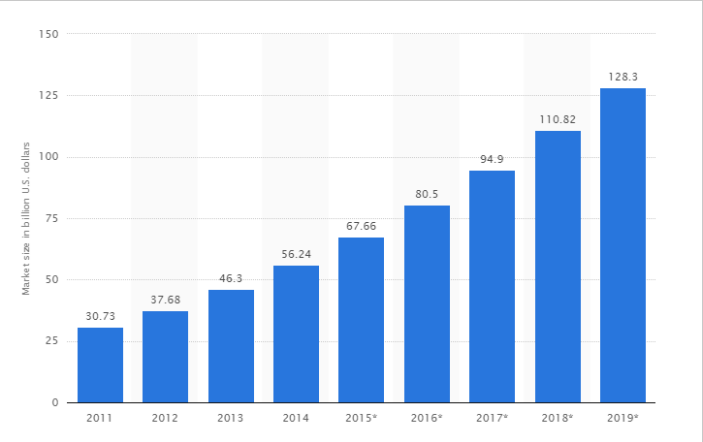
Soutenue devant le jury constitué de :

Mme. Fabienne UZEL-NOUVEL	Rapporteur	INSA/IETR (Rennes)
M. Daniel CHILLET	Rapporteur	ENSSAT/INRIA (Lannion)
M. Pascal BENOIT	Examinateur	UDM2/LIRMM (Montpellier)
M. Bertrand GRANADO	Examinateur	UPMC/LIP6 (Paris)
M. Sébastien PILLEMENT	Examinateur	UDN/IETR (Nantes)
M. Alain GREINER	Directeur de thèse	UPMC/LIP6 (Paris)
M. Quentin MEUNIER	Encadrant de thèse	UPMC/LIP6 (Paris)
M. Franck WAJSBÜRT	Encadrant de thèse	UPMC/LIP6 (Paris)

Introduction

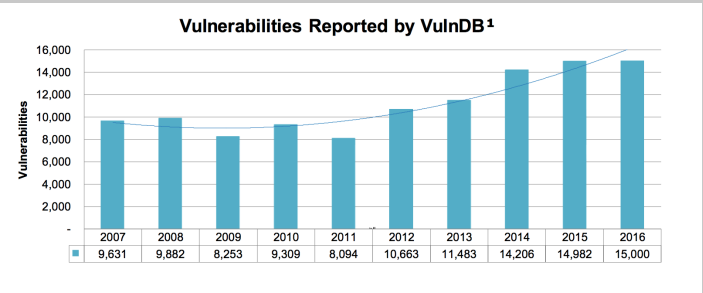
- Augmentation du volume de données numériques et nécessité de garantir leur sécurité
- Augmentation de la puissance de calcul et apparition des architectures fortement multicoeurs
- Possibilité d'utiliser la virtualisation pour optimiser l'utilisation des multiples ressources physiques

- Apparition du *Cloud Computing* qui utilise massivement la virtualisation
- Permet à un utilisateur d'accéder à des ressources distantes
- Marché du *Cloud Computing* en plein essor
- Problématique sur la confidentialité et l'intégrité concernant les données circulant sur ces plateformes



***Source** : Statista [Statista,]*

- La virtualisation introduit un problème de sécurité lié au partage des données
- Le nombre de failles augmentent depuis ces dernières années
- Celles-ci sont de plus en plus nombreuses au sein des systèmes d'exploitation et des hyperviseurs



***Source** : Risk Based Security*

Les trois domaines d'évolutions

Nombres de cœurs intégrés

- L'intégration *Very Large Scale Integration* a permis d'augmenter la densité de transistors présents sur un substrat de silicium
- Perte d'intérêt envers les architectures monocœurs
- Apparition d'architectures multicœurs permettant de tirer parti du parallélisme
- Ce parallélisme offre de nombreux défis en terme de scalabilité

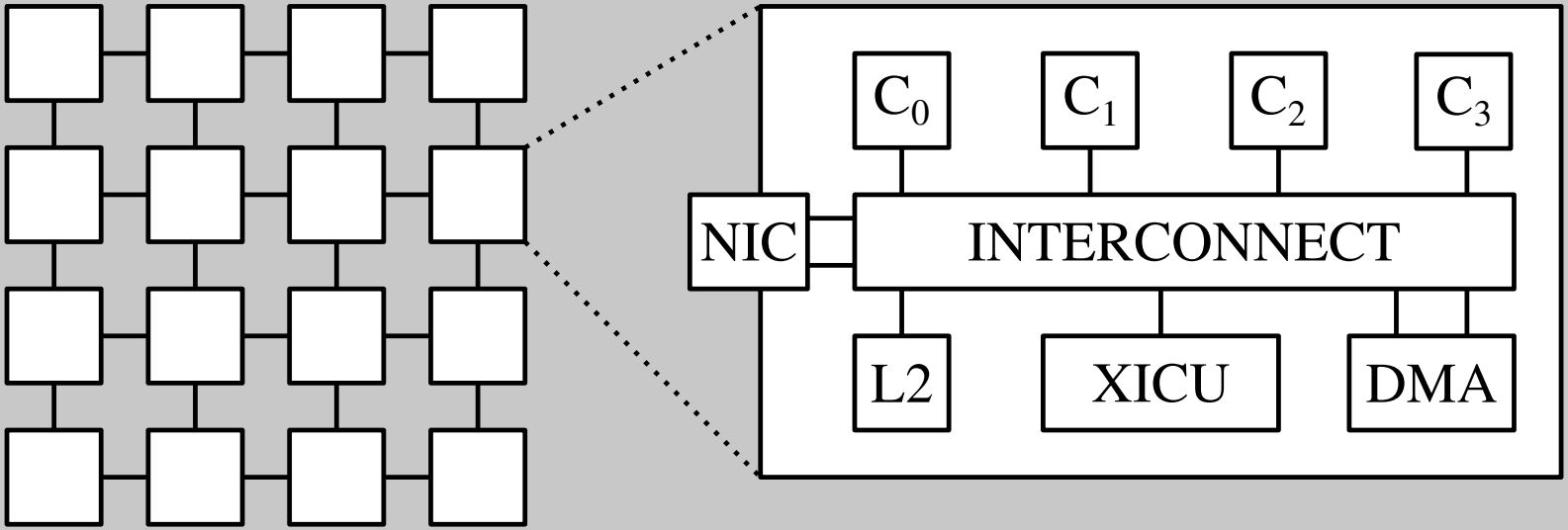
Distribution mémoire

- Évolution d'une distribution centralisée vers une distribution décentralisée pour pallier à un problème de contention
- Apparition du concept de logiquement partagée mais physiquement distribuée
- Deux méthodes : *Symmetric Multi Processor* et *Non Uniform Access Memory*

Cache et cohérence

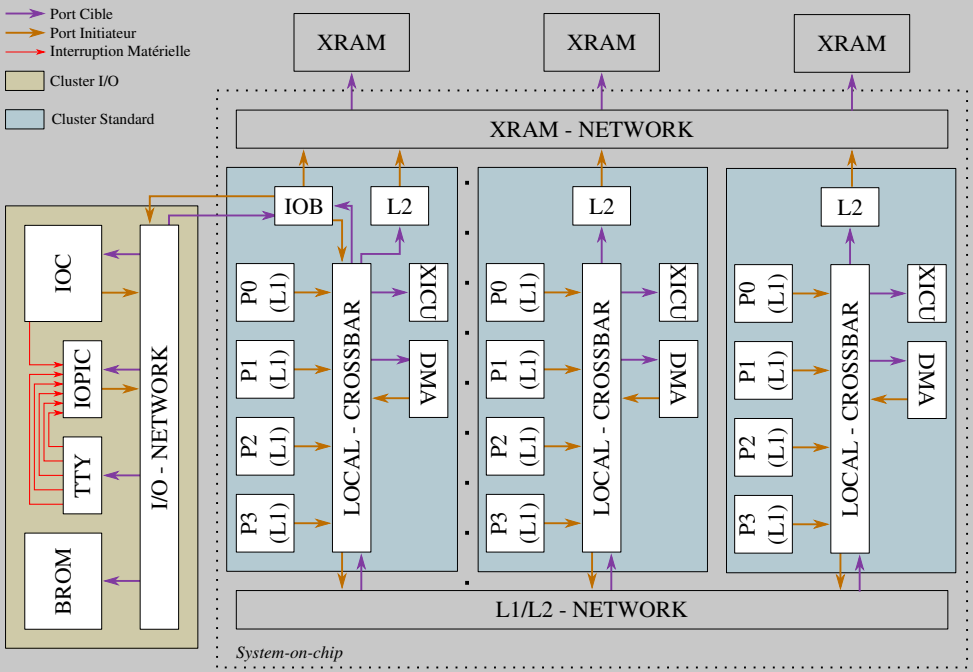
- Évolution des hiérarchies de cache permettant d'augmenter le débit entre les cœurs et la mémoire
- Avec l'apparition des architectures multicœurs la cohérence des caches est devenue un problème important

Architecture manycore



- Les architectures manycore contiennent de plusieurs dizaines à plusieurs centaines de cœurs intégrés sur la même puce
- Ces architectures utilisent des cœurs simples pour permettre de maximiser le ratio performance par Watt consommé [Woo and Lee, 2008].
- Elles sont typiquement clusterisées, et les clusters sont interconnectés entre-eux à l'aide d'un réseau sur puce.
- Chaque cluster contient généralement un nombre variable de coeurs et quelques périphériques.

L'architecture TSAR



- Tous les clusters contiennent :
  - 4 cœurs MIPS avec leurs caches de premier niveau (L1)
  - Un cache de second niveau (L2)
  - 2 périphériques internes : XICU, DMA
  - Un *crossbar* local
- Le cluster I/O contient :
  - Un contrôleur de terminal (TTY)
  - Un contrôleur de disque (IOC)
  - Un contrôleur d'interruptions programmable (IOPIC)
  - Une ROM de démarrage (BROM)

Concepts de la virtualisation

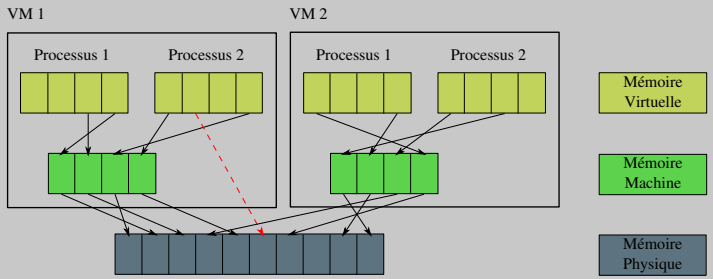
- Inventée dans les années 70, cette technologie a été utilisé sur les serveurs sécurisés permettant d'exécuter plusieurs services en parallèle et de façon sécurisée [Creasy, 1981]
- La virtualisation autorise différentes applications logicielles, développées pour différents systèmes d'exploitation, à s'exécuter sur la même machine physique.
- Une utilisation dans le domaine de la sécurité est l'exécution d'un système d'exploitation non-fiable sous le contrôle d'un logiciel de confiance.
- La virtualisation a introduit deux concepts : la machine virtuelle et l'hyperviseur.
- Il existe deux méthodes de virtualisation : la virtualisation complète et la paravirtualisation.

La virtualisation s'appuie fortement sur les hyperviseurs, mais ceux-ci deviennent de plus en plus complexes et peuvent contenir des failles de sécurité.

Virtualisation du processeur et de la mémoire

- Les techniques de virtualisation du processeur varient en fonction du type de virtualisation utilisé :
  - Virtualisation complète :
    - les instructions privilégiées exécutées par les machines virtuelles sont piégées puis émulées par l'hyperviseur
    - Les instructions sensibles non privilégiées ne peuvent pas être piégées automatiquement
    - Utilisation de la technique *binary translation* pour analyser le code et remplacer ces instructions par des instructions privilégiées
  - Paravirtualisation :
    - L'utilisation d'instructions sensibles est prohibée en modifiant le code noyau du système d'exploitation invité
    - Ces instructions sont remplacées par des appels à l'hyperviseur

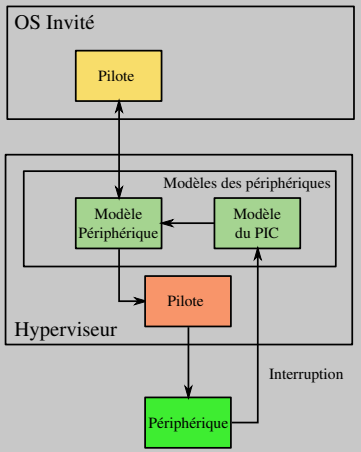
- La virtualisation de la mémoire est très semblable au principe de mémoire virtuelle
  - Un niveau d'espace d'adressage supplémentaire est ajouté
  - L'hyperviseur est en charge du *mapping* de la mémoire physique de la machine virtuelle sur la mémoire de la machine hôte
  - La machine virtuelle est en charge du *mapping* de la mémoire virtuelle sur la mémoire physique de la machine virtuelle



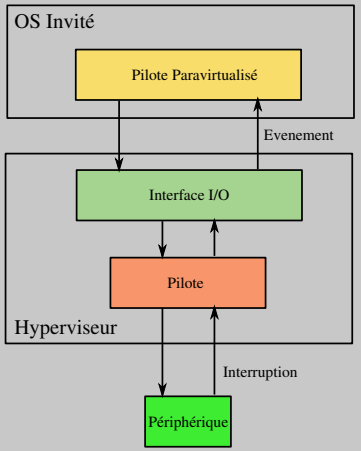
Source : *[Marshall, 2007]*

Virtualisation des périphériques

- Trois principales techniques de virtualisation de périphériques :
  - Émulation :
    - Implémentation du matériel de façon strictement logicielle
    - L'hyperviseur propose aux machines virtuelles un modèle de périphérique
    - Permet la virtualisation complète
    - Surcoût en performance très important à cause des nombreuses transition par l'hyperviseur
  - Paravirtualisation :
    - Interface entre l'hyperviseur et le système invité
    - Nécessite la modification les pilotes des systèmes invités
  - Assignation directe :
    - Allocation d'un périphérique de façon exclusive à un système invité
    - Nécessite la redirection des interruptions vers le bon système invité
    - Permet la virtualisation complète
    - Nécessite de s'assurer que les périphériques n'accèdent pas à des zones de mémoires non autorisées



Source : *[Abramson et al., 2006]*



Source : *[Abramson et al., 2006]*

Modèle de menace

Propriétés de sécurité

- **Confidentialité** : propriété de sécurité qui garantit qu'une information ne peut pas être accédée en lecture par une entité non autorisée [Bell and LaPadula, 1973].
- **Intégrité** : propriété de sécurité qui garantit qu'une information n'a pas été altérée par une entité non autorisée [Biba, 1977].
- **Disponibilité** : propriété de sécurité qui garantit que les entités autorisées peuvent accéder à un service dans un temps borné et raisonnable [PUB, 2004].

Définition de notre *Trusted Computing Base* (TCB)

- Le matériel est considéré de confiance et les attaques physiques ne sont pas considérées
- Les systèmes d'exploitation invités et l'hyperviseur ne sont pas de confiance
- Toute la plateforme matérielle est donc dans la TCB ainsi que quelques fonctions critiques de l'hyperviseur
- Nous cherchons à garantir la confidentialité et l'intégrité des données des machines virtuelles s'exécutant sur la plateforme

Problèmes identifiés

Cohabitation des machines virtuelles avec garanties d'intégrité et de confidentialité

- Les architectures manycore permettent une utilisation massive de la virtualisation en dédiant des ressources matérielles à différents utilisateurs.
- Il faut alors garantir que les données soient accessibles uniquement par les entités autorisées.
- Il est nécessaire de trouver des mécanismes logicielles et matérielles permettant de garantir une cohabitation sécurisée entre les machines virtuelles

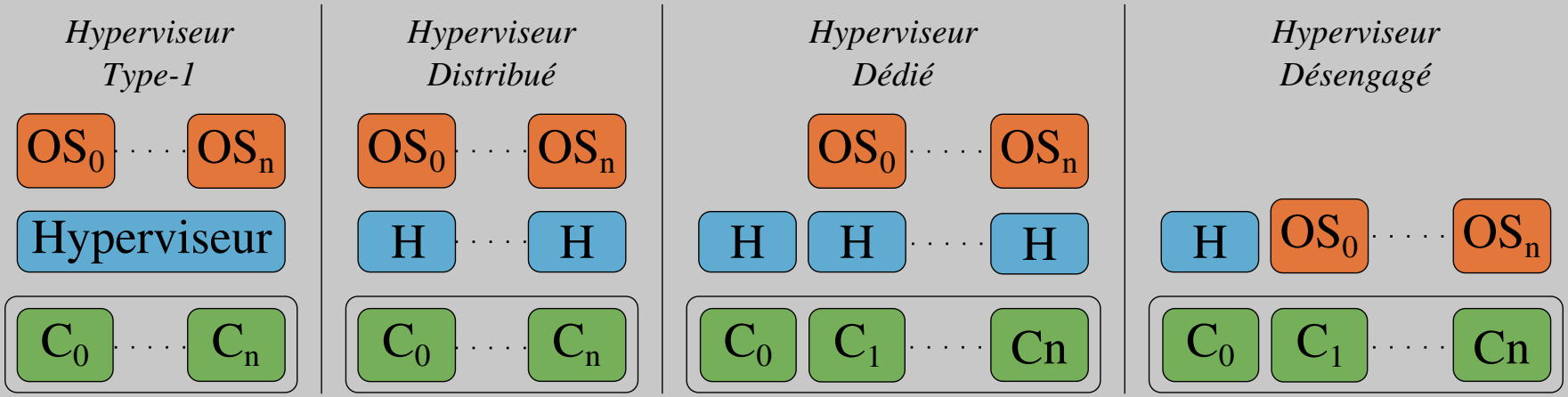
Limitation des droits de l'entité hyperviseur

- Les hyperviseurs deviennent de plus en plus complexe et donc de moins en moins robuste.
- Une faille dans l'hyperviseur met en péril la confidentialité et l'intégrité des logiciels qui s'exécutent sous son contrôle.
- Peut-on alors restreindre l'hyperviseur de sorte qu'il ne puisse pas accéder aux ressources physiques des machines virtuelles ?

Dynamicité de création et de destruction des machines virtuelles

- En mettant en place le concept d'hypervision en aveugle les interactions entre les machines virtuelles et l'hyperviseur est impossible.
- La question du démarrage et de l'arrêt des machines virtuelles devient alors un problème.

Les différents types d'hyperviseurs

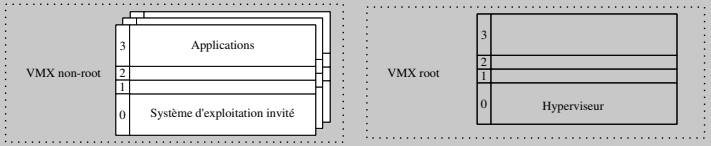


- **Hyperviseur T1** : une seule instance d'hyperviseur gère toutes les ressources, les alloue et qui interagit fortement avec les systèmes d'exploitation invités [Pratt et al., 2005].
- **Hyperviseur distribué** : plusieurs instances d'hyperviseur se partagent la gestion de l'architecture, se base sur le modèle des systèmes d'exploitation multi-kernel [Nightingale et al., 2009].
- **Hyperviseur dédié** : hyperviseur s'exécutant sur des coeurs dédiés et qui gère les systèmes d'exploitation s'exécutant sur les autres cœurs [Dai et al., 2013].
- **Hyperviseur désengagé** : un type d'hyperviseur dédié qui cherche à limiter les interactions avec les machines virtuelles en exécution [Szefer et al., 2011].

Hyperviseur aveugle : un type d'hyperviseur dédié et désengagé qui offre la garantie qu'il ne possède pas d'accès aux ressources matérielles allouées à une machine virtuelle [Dubrulle et al., 2015].

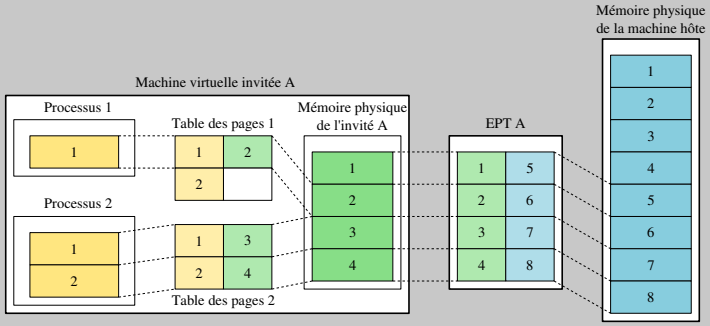
Extensions matérielles pour la virtualisation sécurisée

- Technologies Intel VT-x [Uhlig et al., 2005] et AMD-SVM [Inc., 2005].
- Permet l'exécution d'un système d'exploitation invité dans les mêmes niveaux de privilèges que sur une exécution native.
- Ajout d'un nouveau mode dans lequel l'hyperviseur s'exécute.



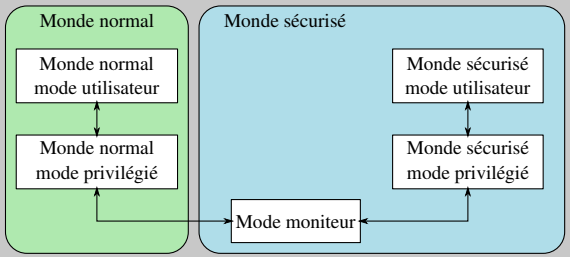
Source : *[Uhlig et al., 2005]*

- Technologies Intel *Extended Page Table* [Neiger et al., 2006] et AMD *Nested Paging* [Inc., 2008]
- Permet de réduire le surcoût lié à la gestion logicielle de la table des pages de la machine virtuelle.
- Le système invité gère les *mappings* entre les pages virtuelles (VPN) et les pages physiques du système invité (MPN)
- L'hyperviseur gère les traductions entre les MPN et les pages physiques du système hôte (PPN)



Source : *[Neiger et al., 2006]*

- Technologie ARM *Trustzone* [Alves and Felton, 2004]
- Cohabitation de deux mondes au sein d'une même plateforme : sécurisé et non sécurisé
- Le bit *Non-Secure Bit* permet de désigner dans quel monde est le processeur à un instant donné.



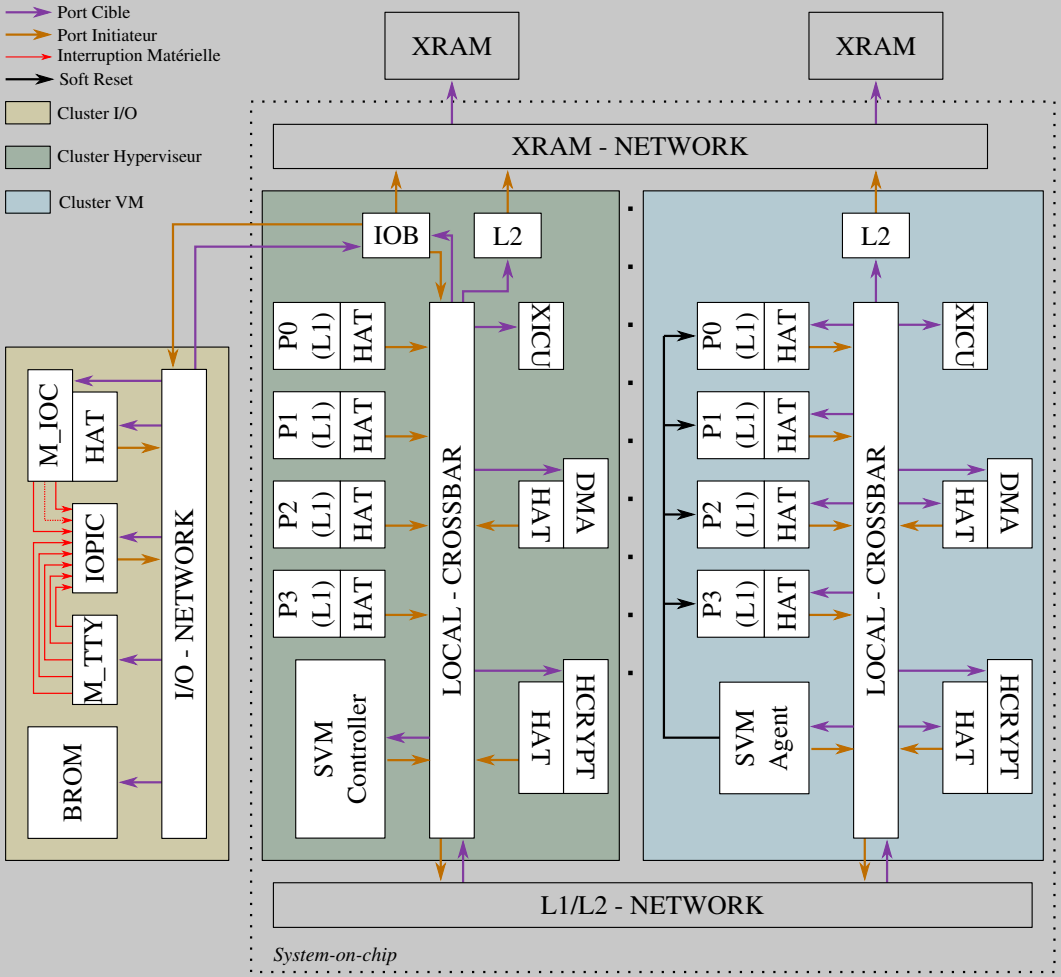
Architectures sécurisées existantes

Nom	Complexification des cœurs	Modifications logicielles	Partage de ressources	TCB	Limitation hyperviseur
Bastion [1]	Oui	Apps. + OS	Oui	CPU + Hyp.	Non
NoHype [2]	Oui	OS	Oui	CPU + Firmware	Pas d'hyp.
HyperWall [3]	Oui	Hyp.	Oui	CPU	Oui
H-SVM [4]	Oui	Hyp. + OS	Oui	CPU	Oui
SEMA [5]	Non	Aucune	Non	CPU + Hyp.	Non
Intel-SGX [6]	Oui	Apps. + OS	Oui	CPU	Oui

Aucunes des solutions présentées ci-dessus répondent à l'ensemble des critères désirés

[1] [Champagne and Lee, 2010]
[2] [Keller et al., 2010]
[3] [Szefer and Lee, 2012]
[4] [Jin et al., 2015]
[5] [Masti et al., 2015]
[6] [Costan and Devadas, 2016]

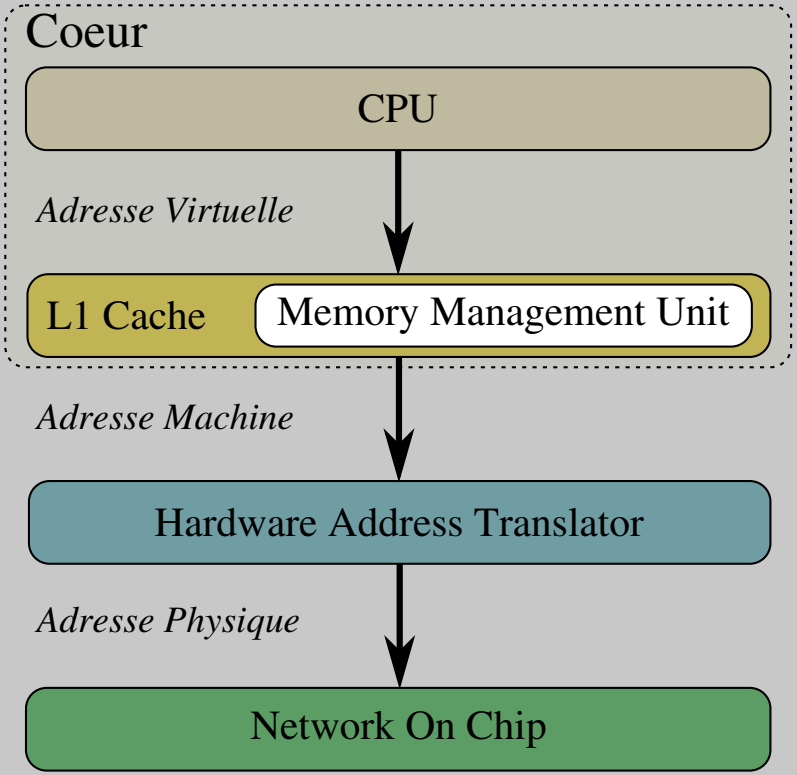
Architecture Tsunamy



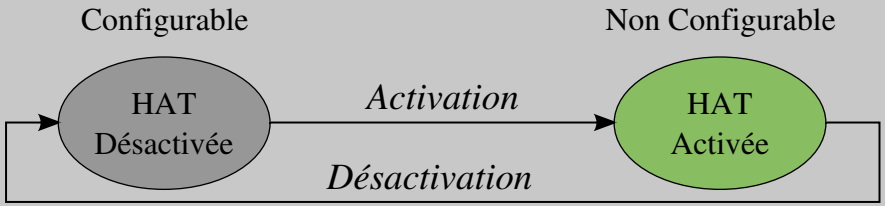
- Tous les clusters contiennent :
  - 4 cœurs MIPS avec leurs caches de premier niveau (L1)
  - Un cache de second niveau (L2)
  - 2 périphériques internes : XICU, DMA
  - Un *crossbar* local
  - Un crypto-processeur (HCRYPT)
  - Les *Hardware Address Translator* (HAT)
  - Les *Shutdown Virtual Machine* (Controller / Agent)
- Le cluster I/O contient :
  - Un contrôleur de terminal (M_TTY)
  - Un contrôleur de disque (M_IOC)
  - Un contrôleur d'interruptions programmable (IOPIC)
  - Une ROM de démarrage (BROM)

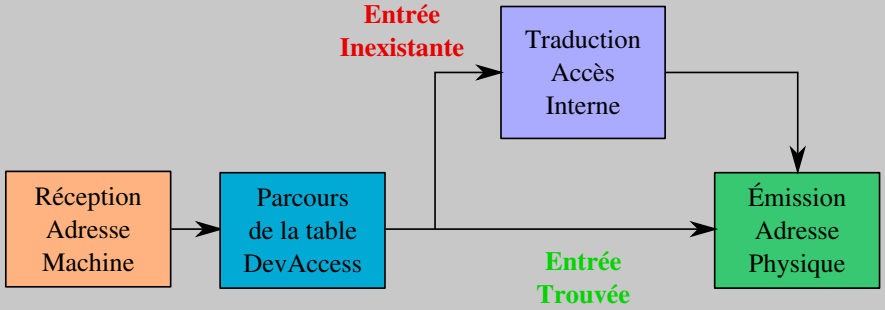
Ajout espace d'adressage

- Isolation physique réalisée par le rajout d'un espace d'adressage :
  - Les adresses machine
- La traduction des adresses machines en adresses physiques assurent que celles-ci ne pourront cibler que des ressources allouées à la machine virtuelle
- La traduction est réalisée à l'aide d'un module de segmentation simple et rapide, qui est répliqué devant chaque initiateur sur le réseau sur puce
  - Les *Hardware Address Translator*



Fonctionnement global d'un HAT

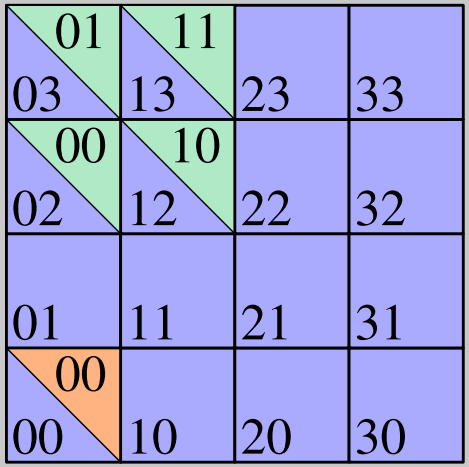


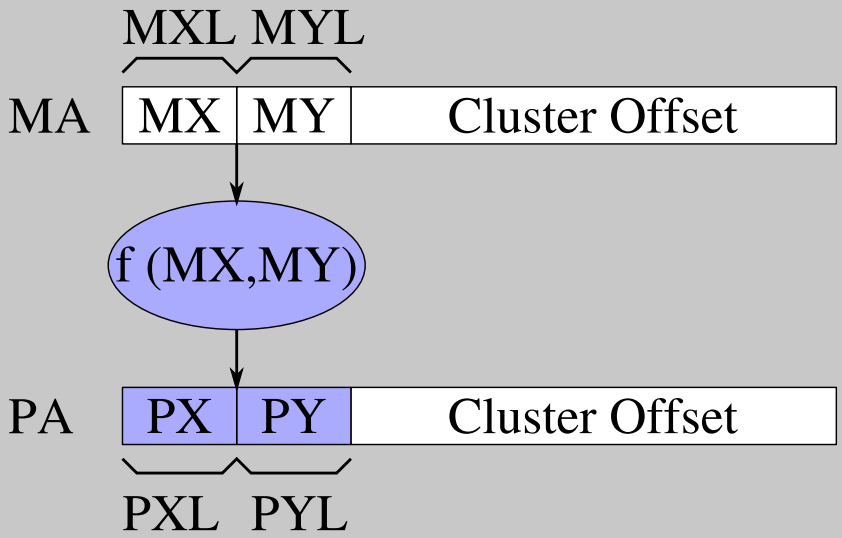


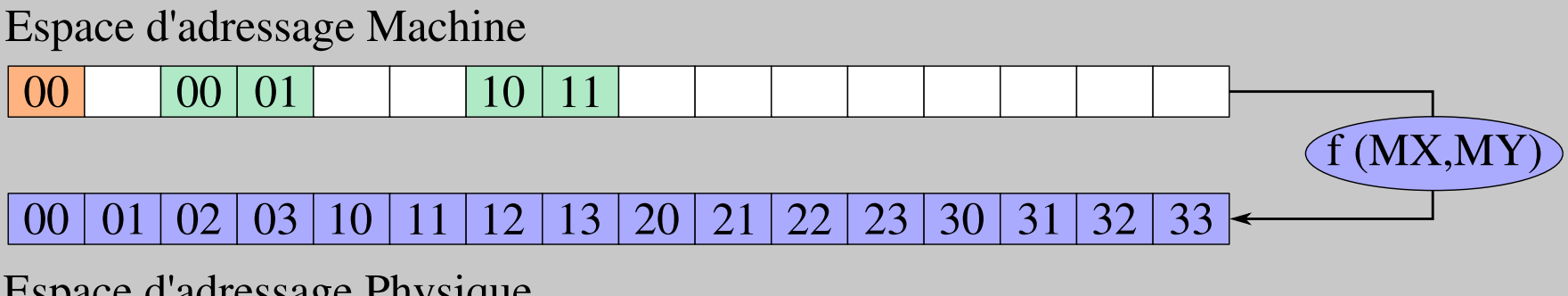
- Un HAT réalise la traduction des adresses machine vers des adresses physiques
- Un HAT opère avec une granularité gros grains (granularité cluster)
- Configuré une fois par l'hyperviseur au démarrage d'un système d'exploitation invité et placé devant chaque initiateur dans l'architecture
- Un HAT a seulement besoin des informations de topologie pour réaliser la traduction d'adresse
- Le mécanisme de traduction dure seulement 1 cycle

- Deux types d'adresses :
  - Un module inclus dans un cluster de la même machine virtuelle (accès interne)
  - Un périphérique externe (accès externe)
- Un accès interne ne peut pas échouer et sa traduction cible toujours une adresse physique située à l'intérieur d'un cluster de la machine virtuelle

Mécanisme des traductions des accès internes

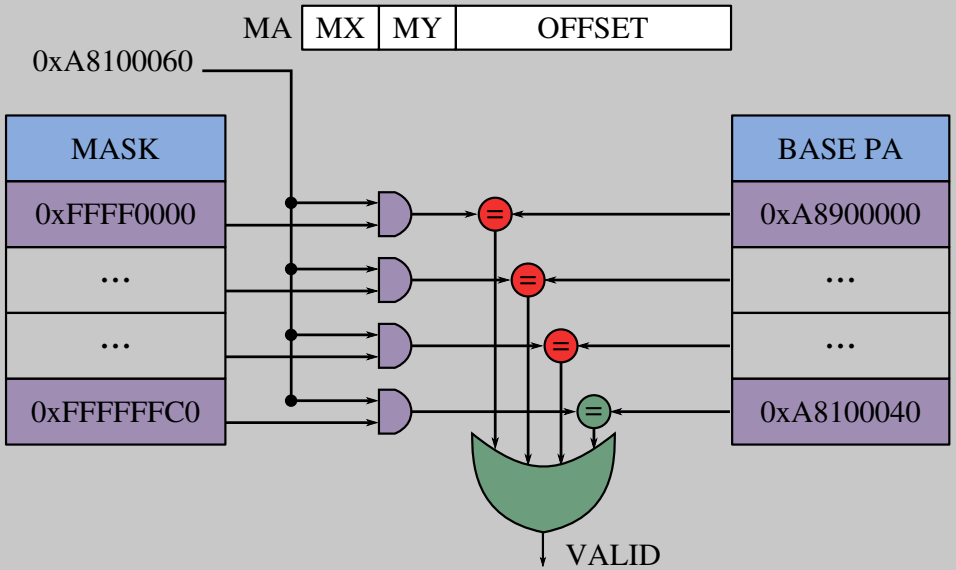




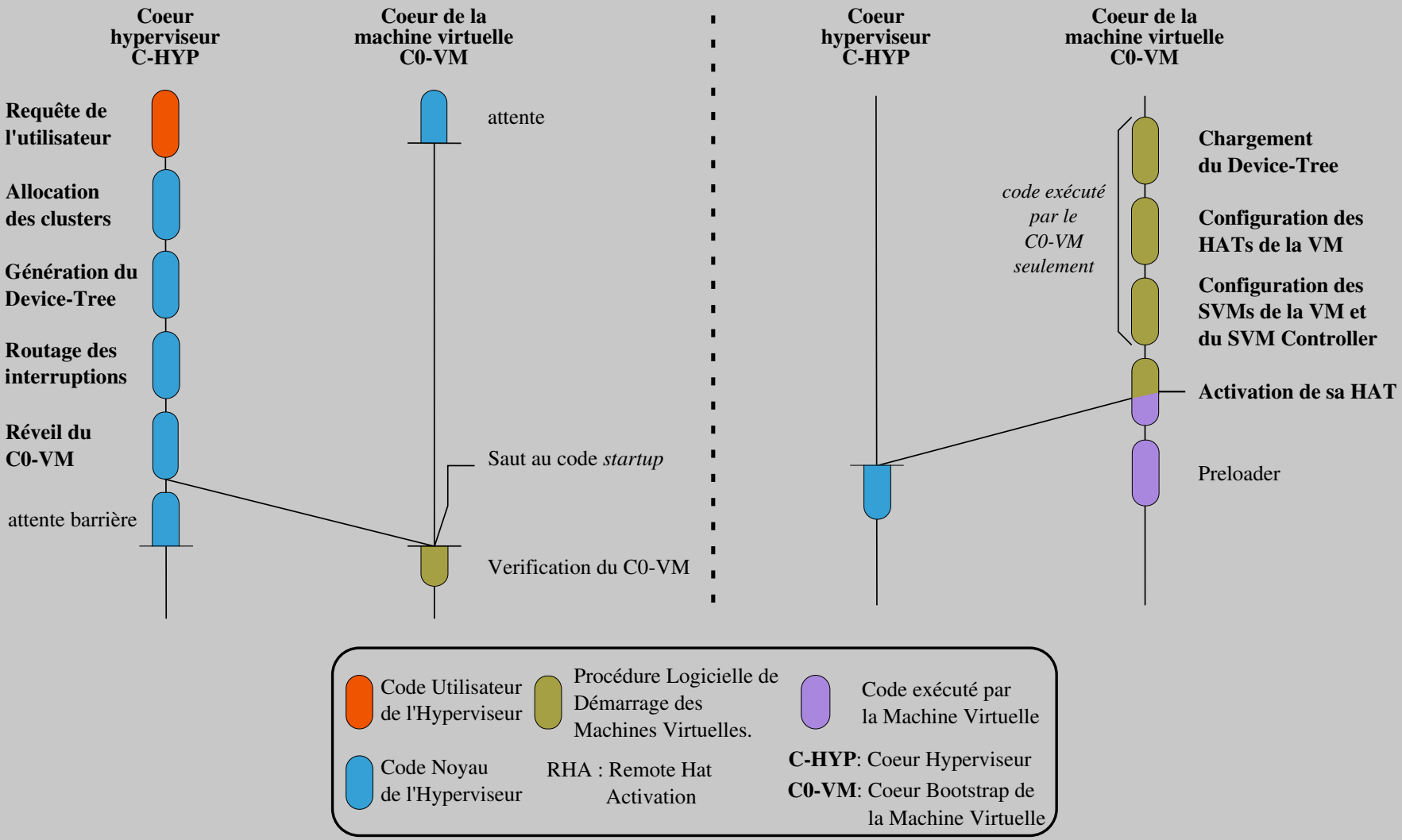


Mécanisme des traductions des accès externes

- 2 tables permettent de gérer les accès aux périphériques :
  - La table contenant l'adresse physique de base des segments autorisés (BASE PA)
  - La table des masques (MASK)
- Permet de vérifier la validité d'un accès avec un coût matériel très faible
- Mécanisme similaire à la segmentation



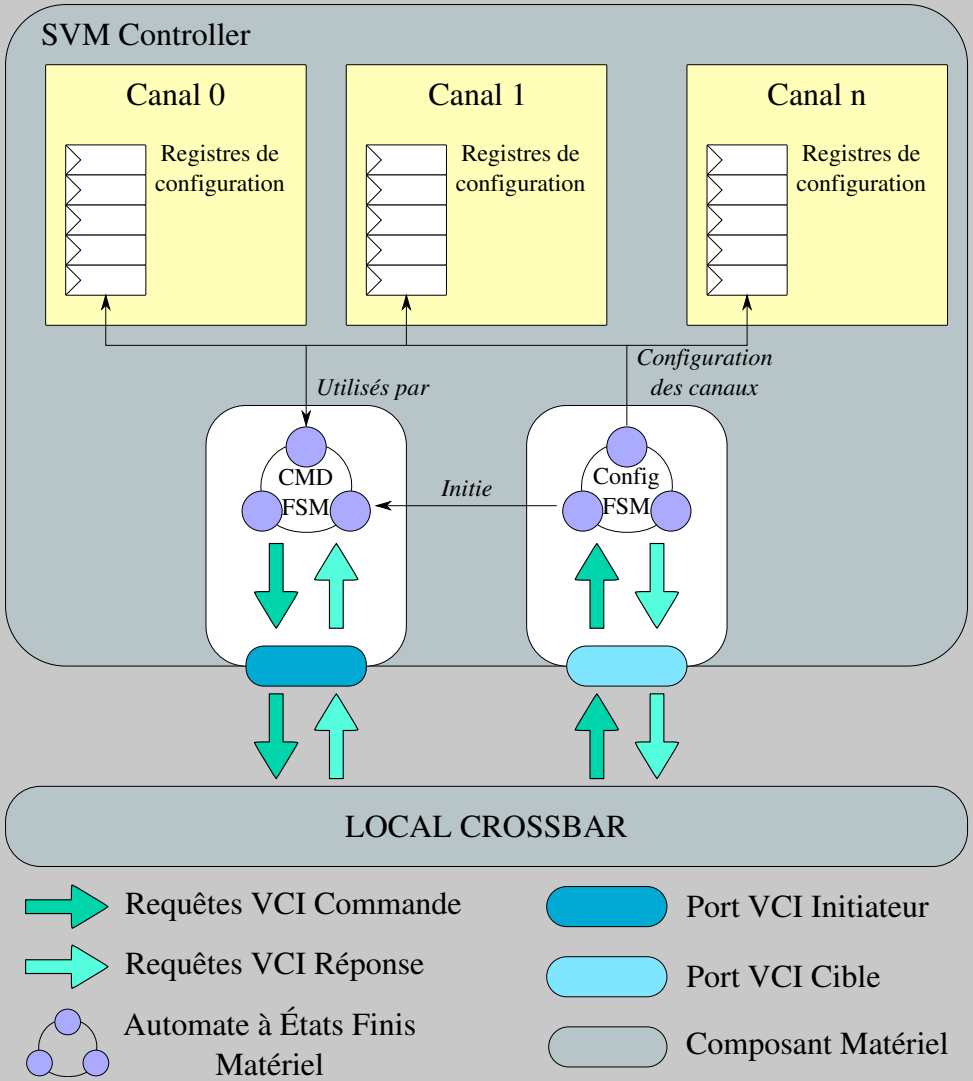
Procédure de démarrage d'une machine virtuelle



Remote HAT Activation

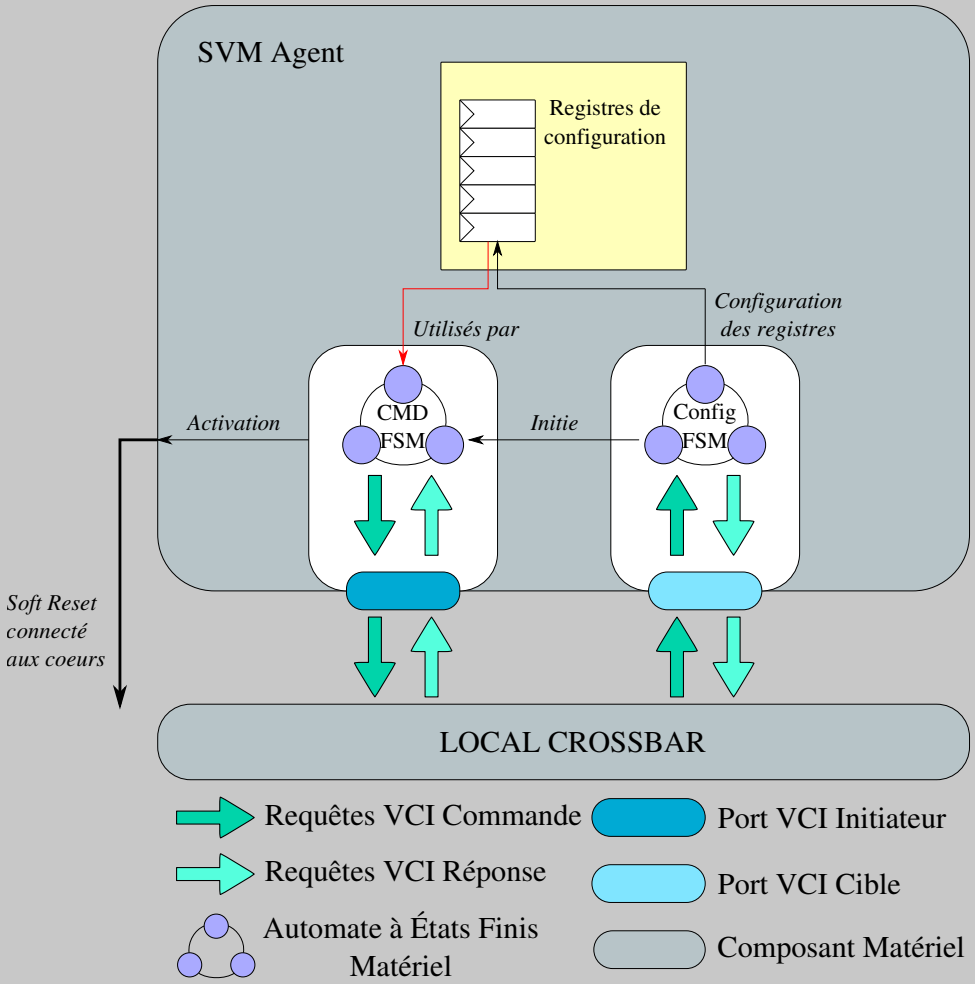
```
1     # t0 contient les 8 bits de poids fort de l'adresse des HATs
2     # t1 contient les 32 bits de poids faible de l'adresse des HATs
3     # t2 contient le mode souhaité pour les HATs
4     # t3 contient l'adresse du point d'entrée
5     # t4 contient l'adresse de la description de l'architecture
6     # t5 contient l'adresse du TTY alloué à la machine virtuelle
7     # t6 contient l'adresse de l'IOC alloué à la machine virtuelle
8     # t7 contient l'adresse du MMC alloué à la machine virtuelle
9
10    sync
11    nop
12    mtc2 zero, $1, 0
13    mtc2 zero, $3, 0
14    mtc2 zero, $2, 0
15    move a0, t7
16    move a1, t5
17    move a2, t6
18    move a3, t4
19    mtc2 t0, $24, 2
20    sw   t2, 0(t1)
21    mtc2 zero, $24, 2
22    sync
23    nop
24    li   $24,  3
25    jalr t3
26    mtc2 $24, $1
27
```

Shutdown Virtual Machine Controller



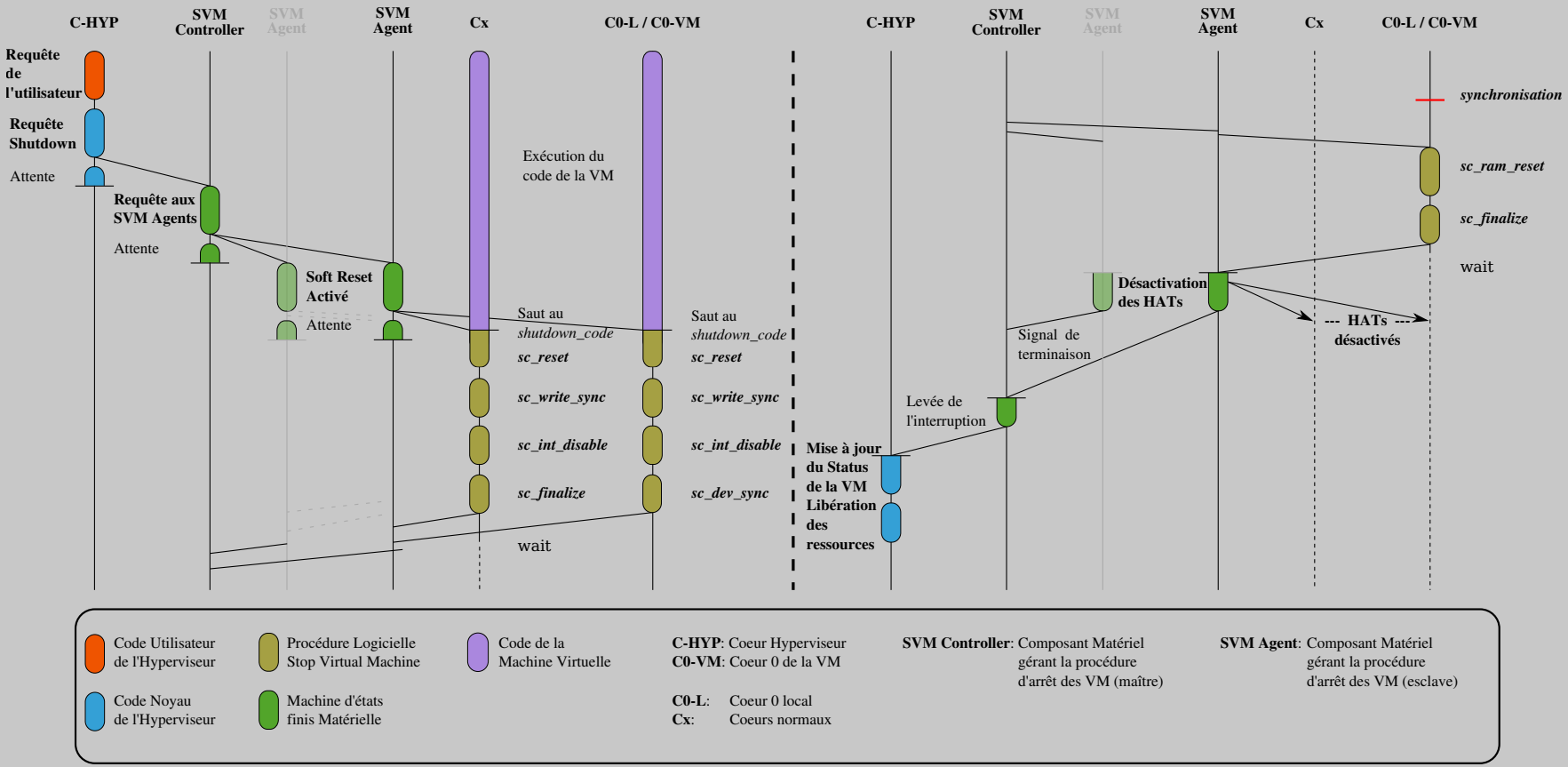
- Composant maître vu par l'hyperviseur
- Composant unique situé dans le cluster hyperviseur
- Composant en charge d'initier la procédure d'arrêt des machines virtuelles
- Contient un nombre de canaux égale au nombre de machines virtuelles pouvant être exécutées simultanément
- Chaque canal contient un jeu de registre de configuration

Shutdown Virtual Machine Agent

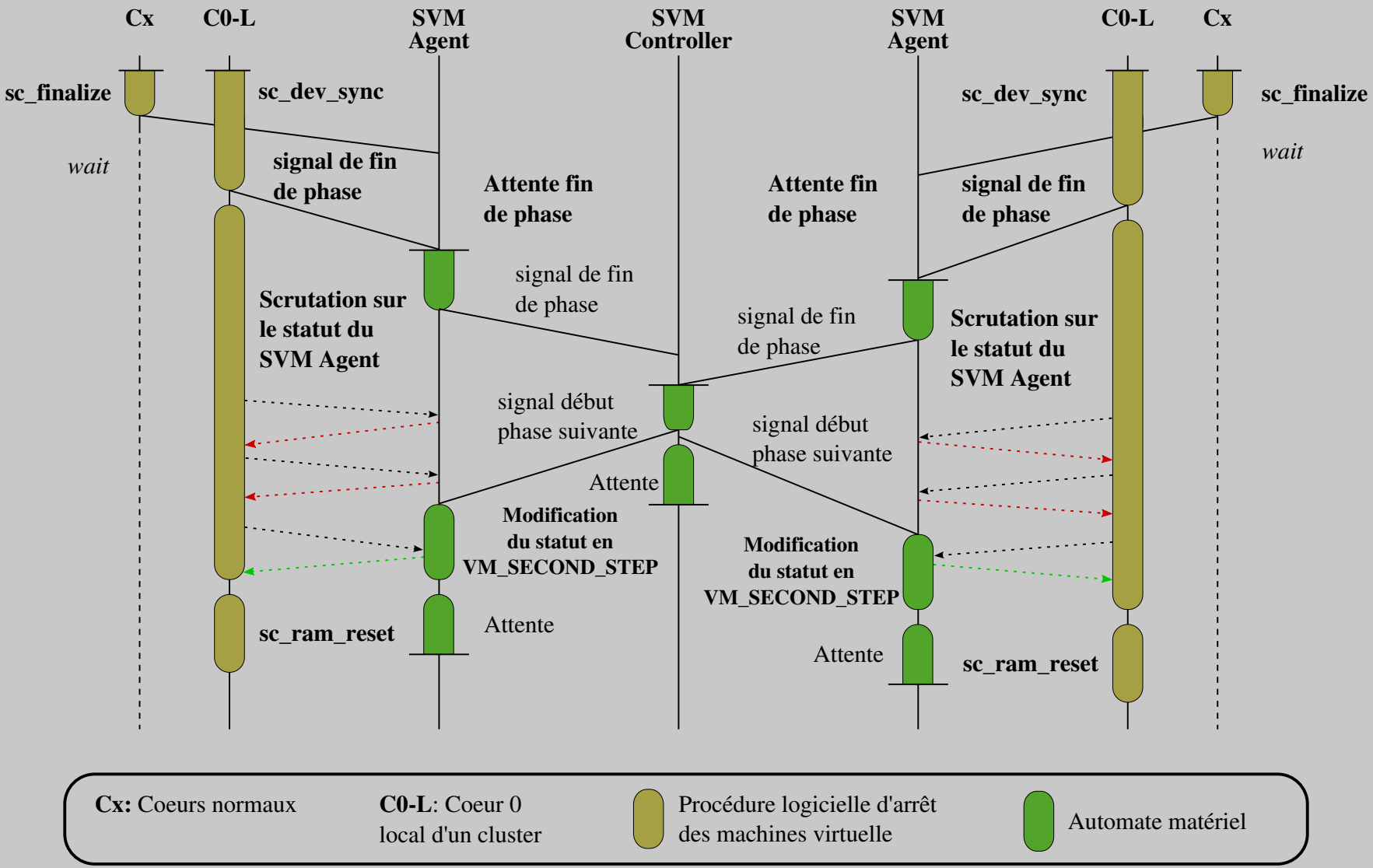


- Composant répliqué dans chaque cluster VM
- Composant en charge d'effectuer la procédure d'arrêt des machines virtuelles
- Chaque SVM agent contient un jeu de registre de configuration
- Composant capable de désactiver les HATs contenus dans son cluster
- Possède un signal *soft reset* permettant de faire brancher les cœurs à la procédure logicielle d'arrêt

Procédure d'arrêt d'une machine virtuelle



Synchronisation matérielle



Évaluation du surcoût matériel

HAT

- 182 octets de mémoire par HAT.
- En comparaison : une ligne du cache L1 contient 64 octets de mémoire.
- Coût total en bits mémorisant d'un cache L1 : 32Ko.
- Coût total en bits mémorisant des HATs : 1 092 octets par clusters.

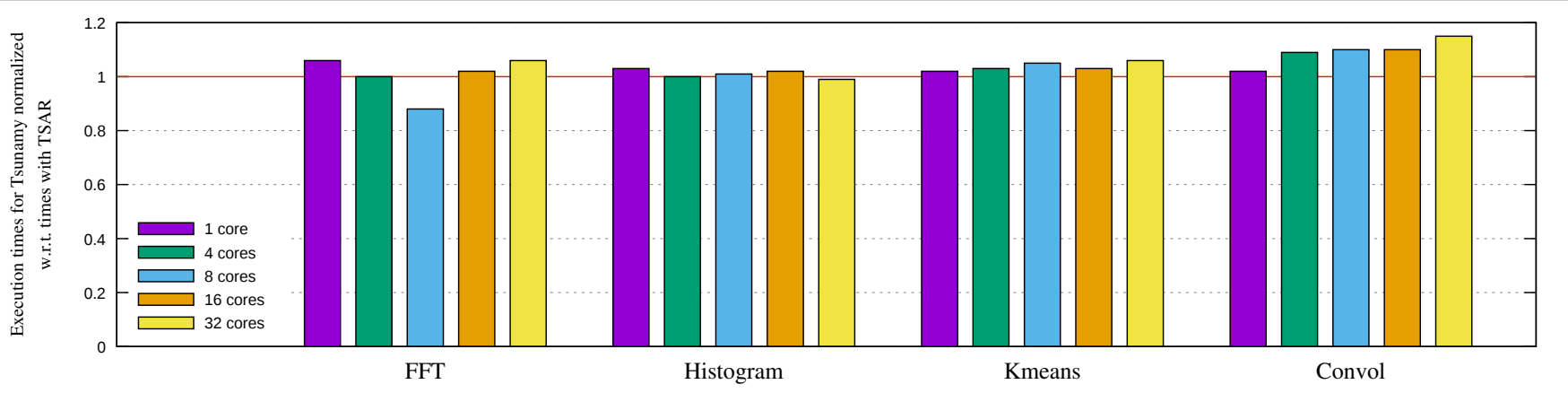
SVM Controller

- 24 octets de mémoire pour le fonctionnement interne.
- 5 octets de mémoire par canal.
- Cas d'étude à 16 canaux : 104 octets de mémoire.

SVM Agent

- 21 octets de mémoire pour son fonctionnement interne.
- 21 octets de mémoire pour les registres de configuration.
- Cas d'étude à 16 clusters : 630 octets de mémoire.

Surcoût temporel induit par les HATs



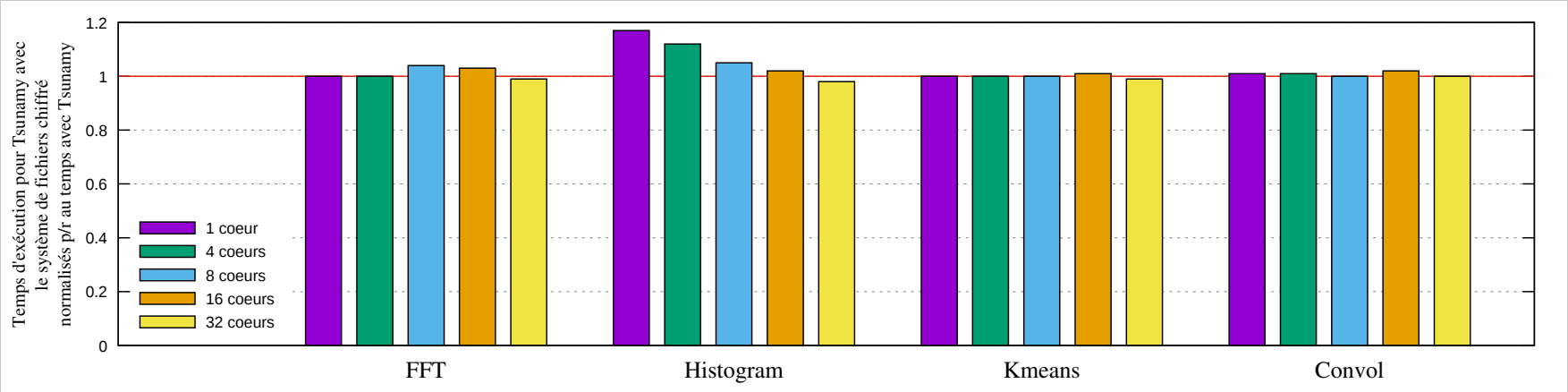
Protocole expérimental

- Chaque application exécuté sur des configurations à 1, 4, 8, 16 et 32 threads
- Un thread par cœur et une seule machine virtuelle déployée
- Temps mesurés sur la phase parallèle des applications

Résultat

- En moyenne 3% de pénalité en performance

Surcoût temporel induit par le système de fichiers chiffré



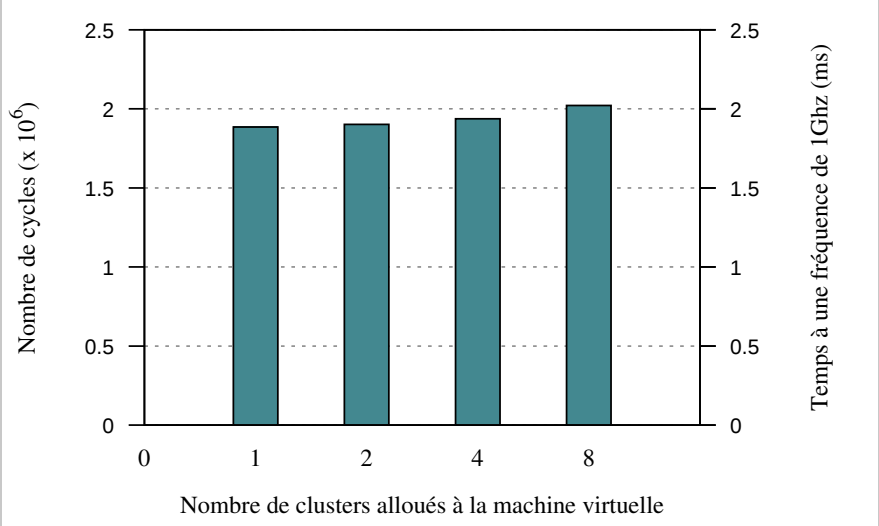
Protocole expérimental

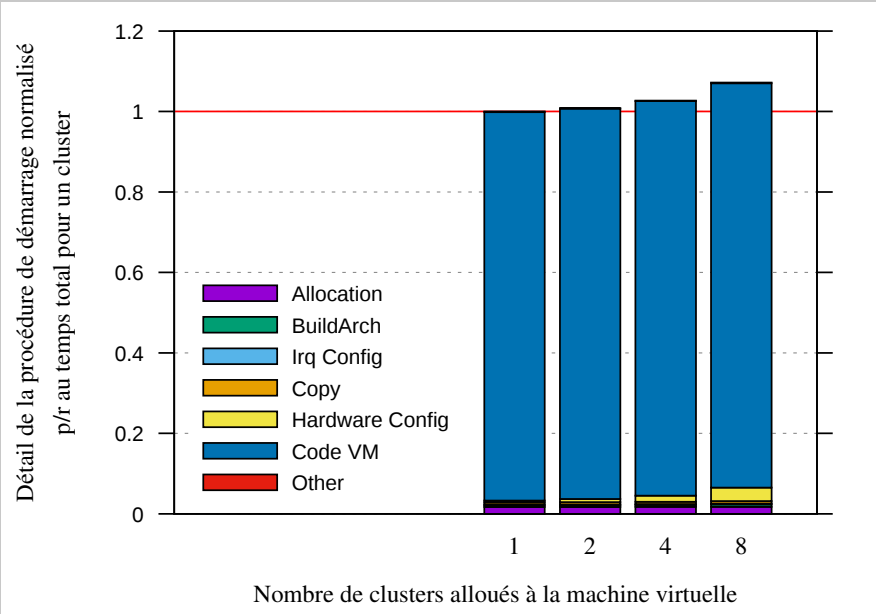
- Chaque application exécuté sur des configurations à 1, 4, 8, 16 et 32 threads
- Un thread par cœur et une seule machine virtuelle déployée
- Temps mesurés sur l'ensemble de l'application

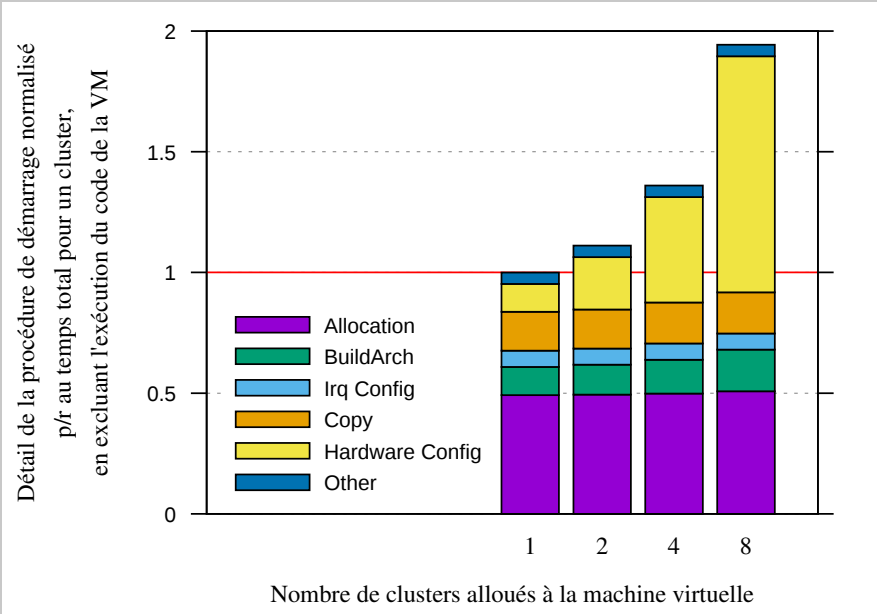
Résultat

- Jusqu'à 18% de pénalité pour une application utilisant de façon intensive le disque

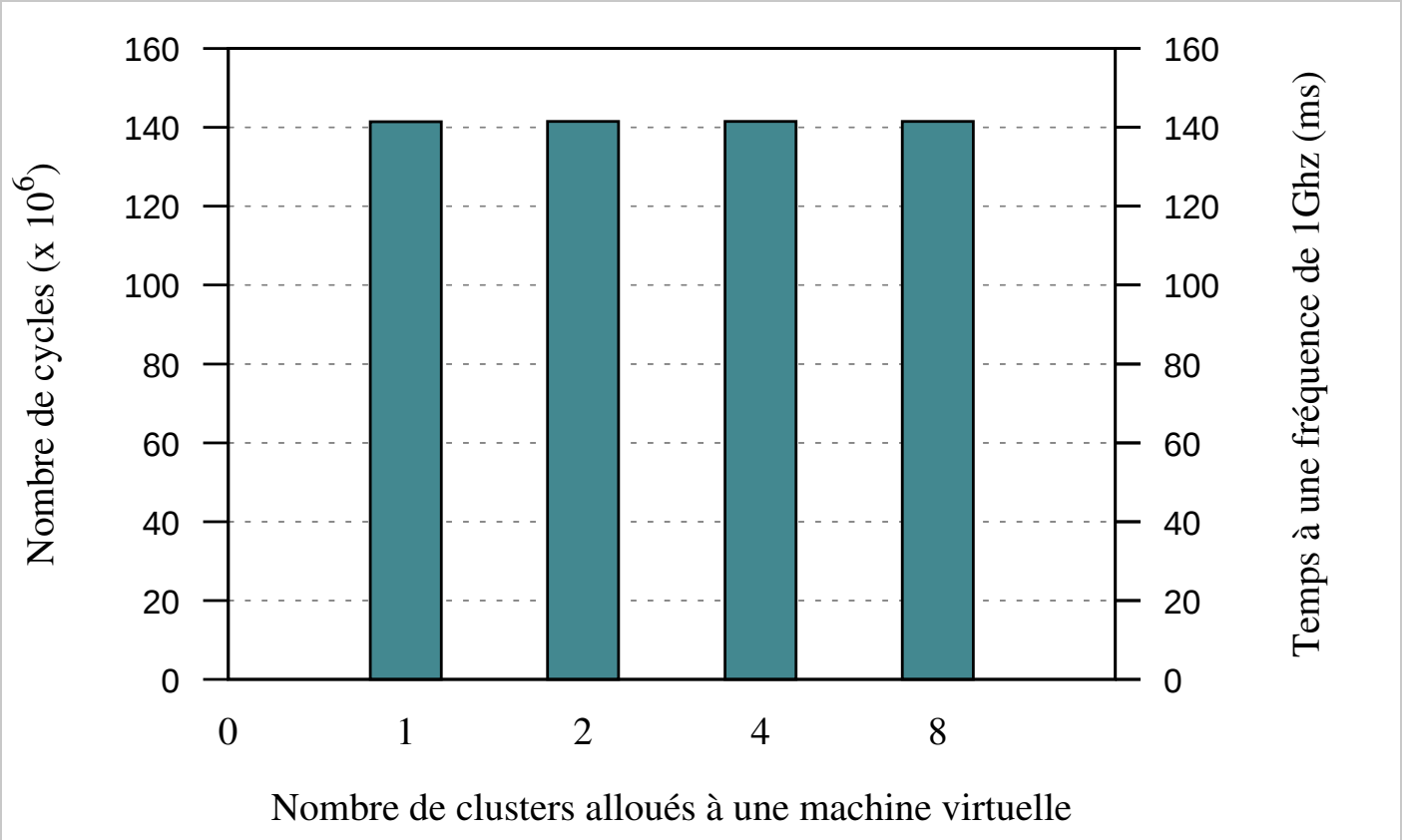
Coût temporel de la procédure de démarrage







Coût temporel de la procédure d'arrêt des machines virtuelles



Évaluation de l'impact de la cohabitation de machines virtuelles

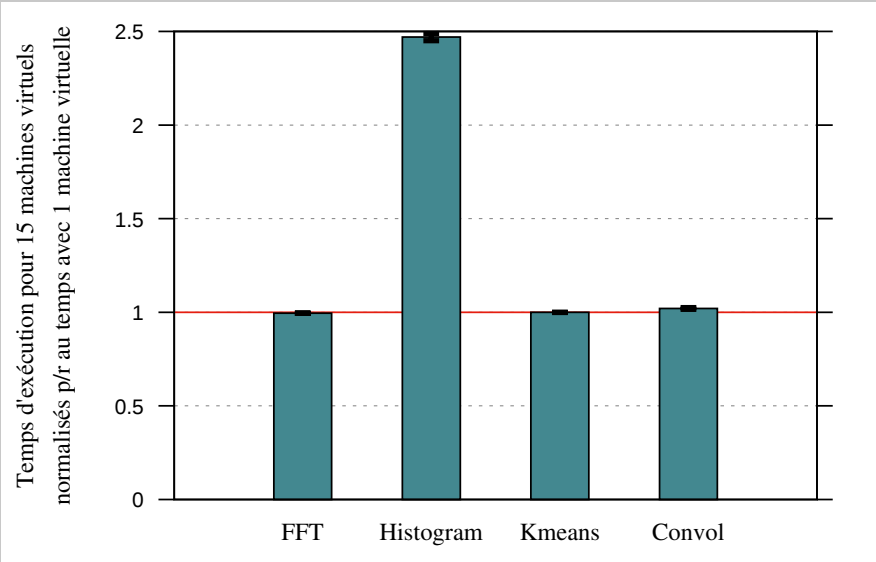


FIGURE: Toutes les phases des applications

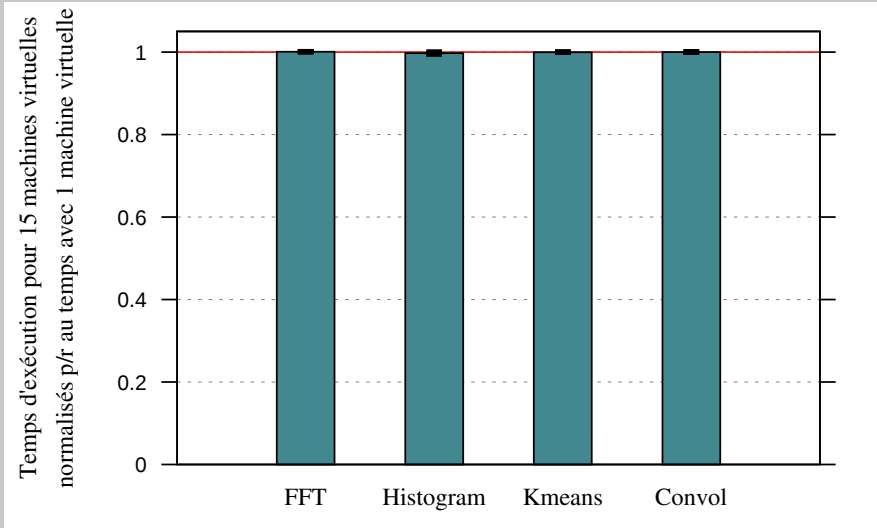
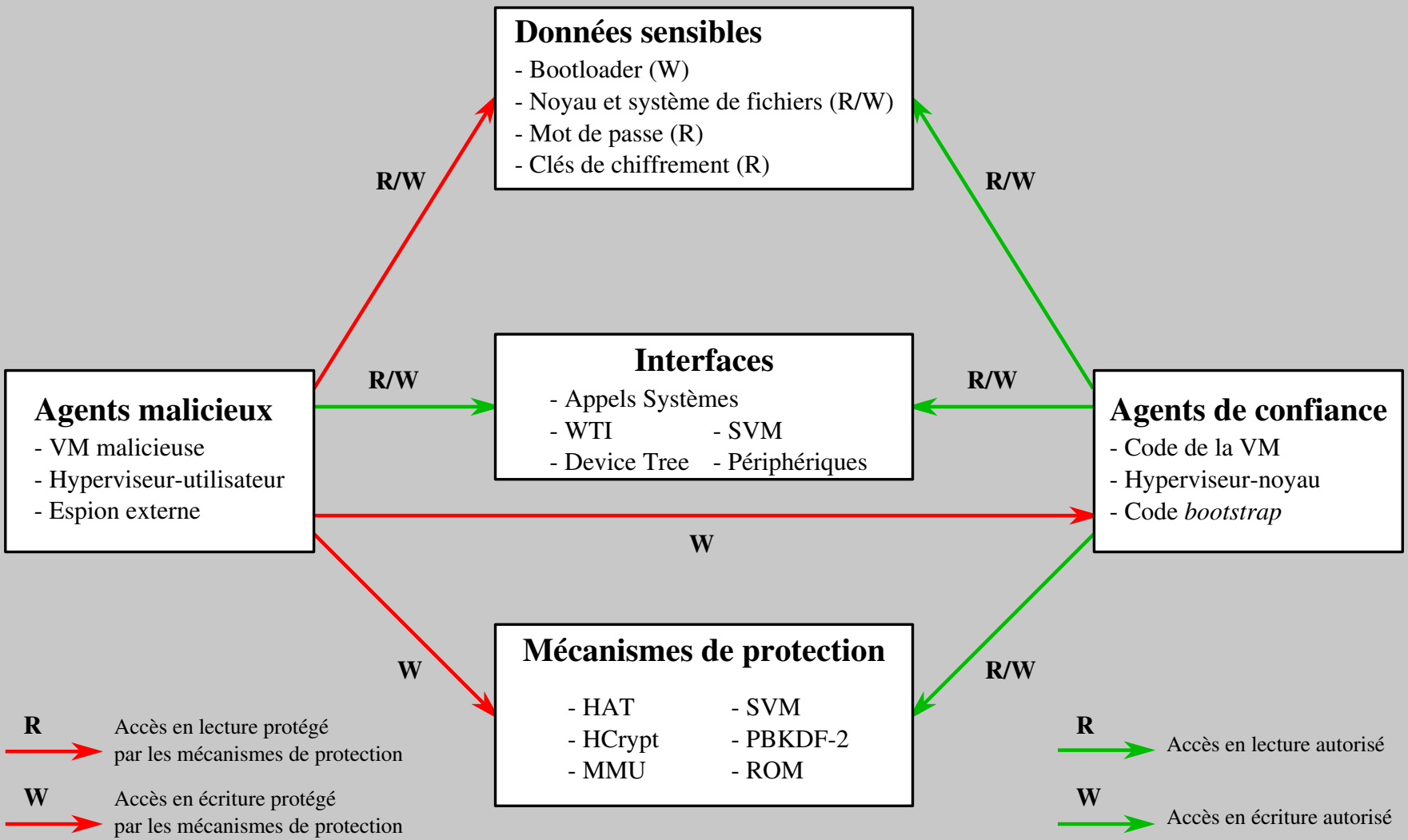


FIGURE: Phase parallèle des applications

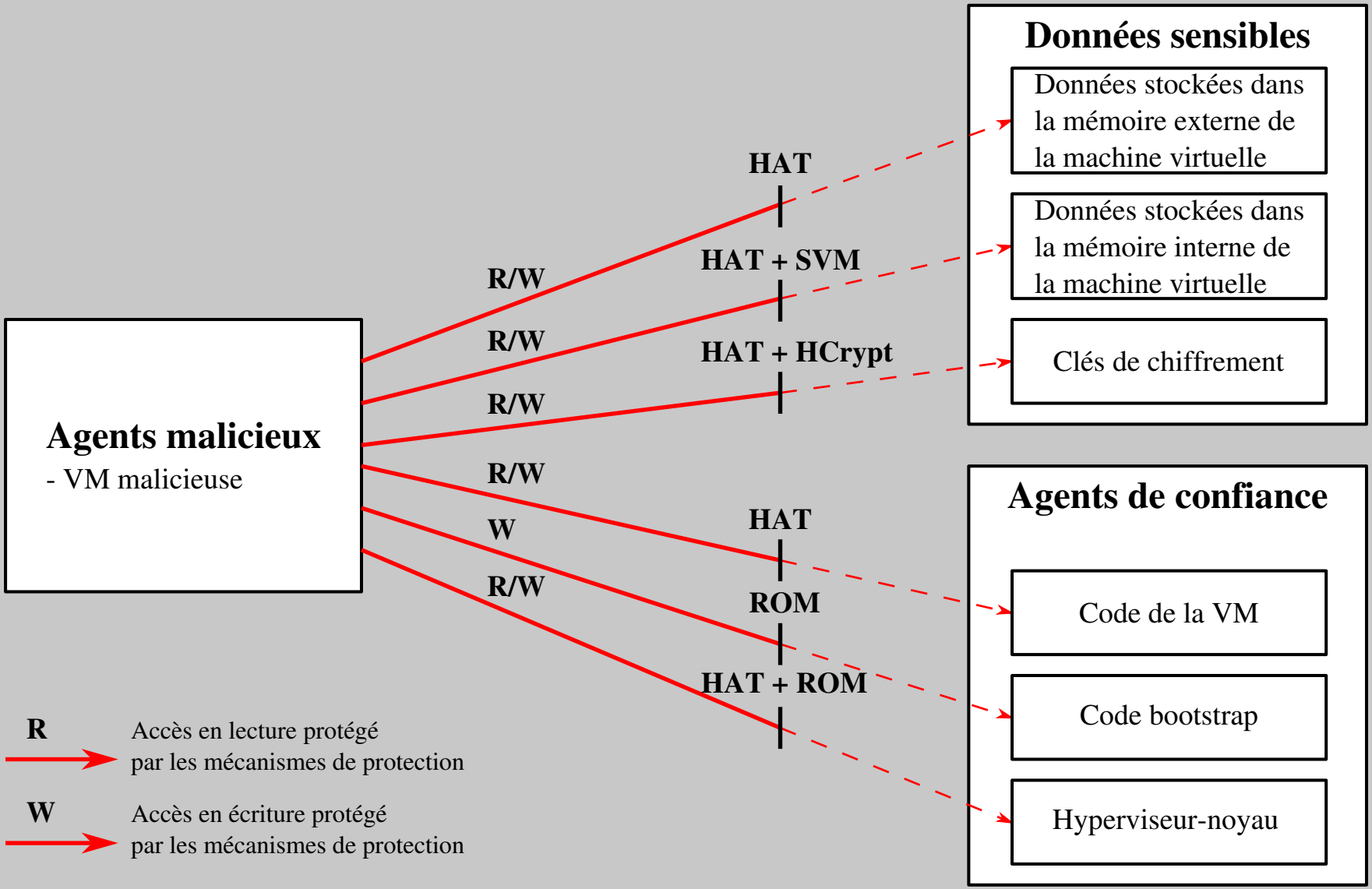
Résultats

- Pas de pénalités pour les applications sans accès disque
- Pénalité importante pour Histogram due à une contention sur le contrôleur de disque

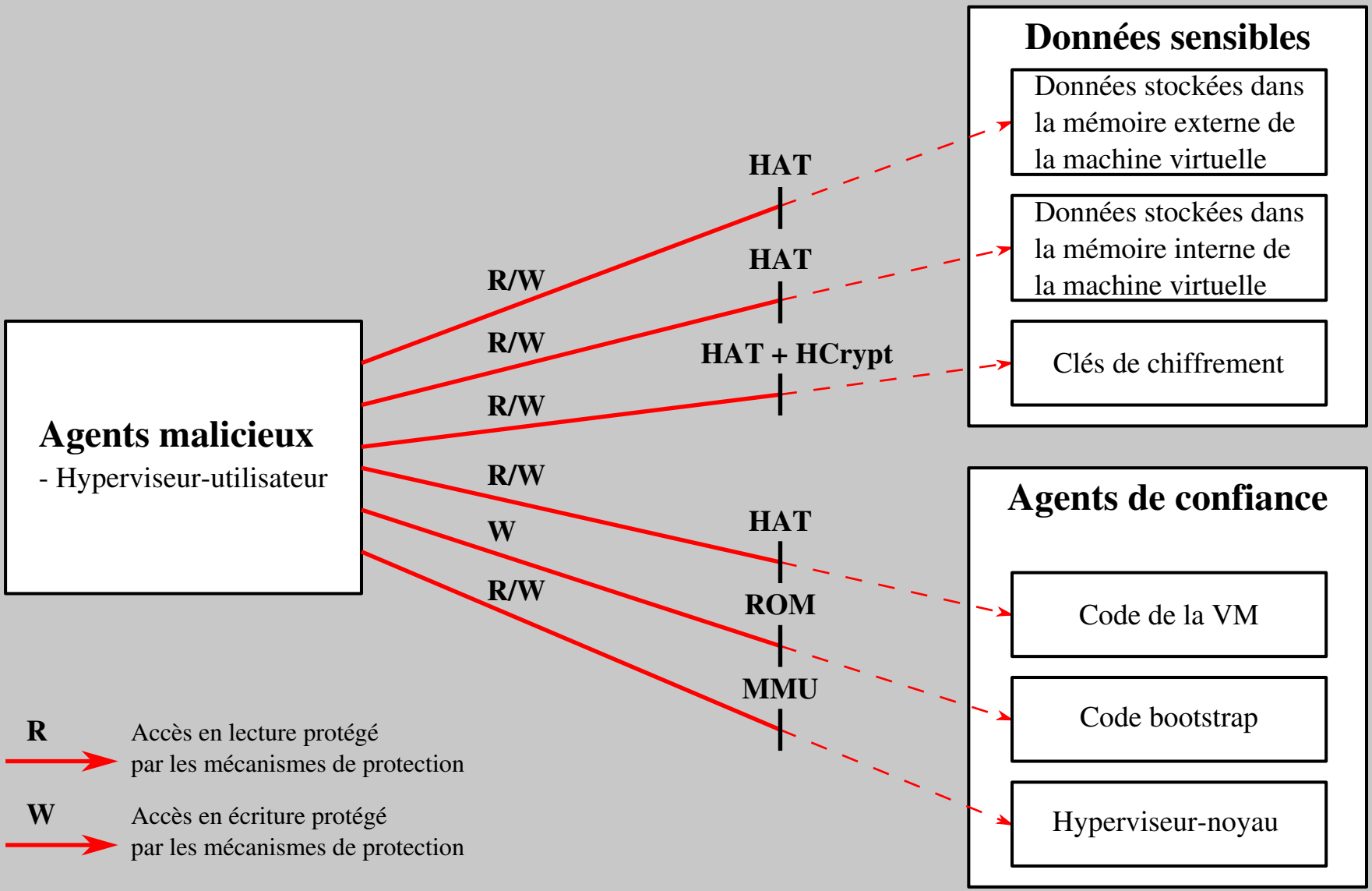
Interactions entre les différents acteurs



Attaques par une machine virtuelle malicieuse



Attaques ciblant la partie utilisateur de l'hyperviseur



Attaques sur les périphériques

Scénario

- Accès physique en lecture ou écriture sur le disque d'une machine virtuelle
- Le noyau et le système de fichiers sont chiffrés
- Une lecture sur le noyau et le système de fichiers ne compromet pas la confidentialité
- Une écriture sur le noyau et le système de fichiers ne compromet pas l'intégrité
- Une écriture peut néanmoins entrainer une perte de données

Faille de sécurité

- Le *bootloader* n'est pas chiffré et il n'y a pas d'authentification de code
- Le remplacement du code du *bootloader* peut mener à une fuite d'informations

Cohabitation de machines virtuelles

- Ajout d'un nouvel espace d'adressage et des composants *Hardware Address Translator* (HAT)
- L'ensemble de ces éléments permet la virtualisation complète et l'isolation physique des machines virtuelles
- Le mécanisme de traduction des HATs utilise des informations de topologie pour les accès internes et une table de segment pour les accès externes
- Le surcoût matériel est inférieur à 1% et le surcoût en performance de l'ordre de 3%

Limitation des droits de l'hyperviseur

- Présence d'HATs spécifiques pour le cluster hyperviseur
- Impossibilité pour l'hyperviseur de sortir de son cluster dédié

Dynamicité de création et de destruction des machines virtuelles

- Mécanisme de démarrage initié par l'hyperviseur mais les étapes concernant la sécurité sont réalisées par les coeurs de la machine virtuelle
- Le code de confiance *startup_code* effectue la configuration et l'activation des HATs
- Mécanisme d'arrêt propose une interface accessible par l'hyperviseur et les machines virtuelles
- Il s'appuie sur deux composants matériels et sur une procédure logicielle
- La procédure d'arrêt garantit qu'une fois accomplie les ressources précédemment allouées sont libérées et ré-utilisables pour une nouvelle machine virtuelle

Perspectives

Authentification du Boot

- Mettre en place un système permettant l'authentification du *bootloader*
- Étude des systèmes basés sur *Trusted Platform Module* (TPM)

Séparation de l'hyperviseur et mécanismes de récupération

- Mettre en place la séparation de la partie noyau et utilisateur de l'hyperviseur
- Mettre en place un système de récupération d'erreur au sein de l'hyperviseur pour éviter qu'un dysfonctionnement ne mène au redémarrage de la machine

Intégration d'une pile IP et infrastructure *cloud computing*

- Rendre réaliste les communications entre les utilisateurs et l'hyperviseur via des communications réseaux
- S'assurer que les solutions existantes pour la sécurisation de ces communications soient compatibles avec le concept d'hyperviseur aveugle

Dissémination des sources et publications

Dissémination des sources

- Code source des composants matériels ainsi que celui de l'hyperviseur disponible sur le dépôt SVN du projet ANR Tsunamy : `https://www-soc.lip6.fr/svn/tsunamy`
- Code sous licence *open-source* : *GNU Lesser General Public License*

Publications

- Deux publications :
  - [Dévigne et al., 2015] : Conférence internationale *2015 Nordic Circuits and Systems Conference* (NORCAS'15)
  - [Dévigne et al., 2017] : Revue *Microprocessors and Microsystems : Embedded Hardware Design* (MICPRO) extension de l'article de la conférence NORCAS'15

Références I

Abramson, D., Jackson, J., Muthrasanallur, S., Neiger, G., Regnier, G., Sankaran, R., Schoinas, I., Uhlig, R., Vembu, B., and Wiegert, J. (2006). Intel virtualization technology for directed i/o. *Intel technology journal*, 10(3).

Alves, T. and Felton, D. (2004). Trustzone : Integrated hardware and software security. *ARM white paper*, 3(4) :18–24.

Bell, D. E. and LaPadula, L. J. (1973). Secure computer systems : Mathematical foundations. Technical report, DTIC Document.

Biba, K. J. (1977). Integrity considerations for secure computer systems. Technical report, DTIC Document.

Champagne, D. and Lee, R. B. (2010). Scalable architectural support for trusted software. In *HPCA - 16 2010 The Sixteenth International Symposium on High-Performance Computer Architecture*, pages 1–12.

Costan, V. and Devadas, S. (2016). Intel sgx explained. *IACR Cryptology ePrint Archive*, 2016 :86.

Creasy, R. J. (1981). The origin of the vm/370 time-sharing system. *IBM Journal of Research and Development*, 25(5) :483–490.

Dai, Y., Qi, Y., Ren, J., Shi, Y., Wang, X., and Yu, X. (2013). A lightweight vmm on many core for high performance computing. SIGPLAN Not.

Dubrulle, P., Sirdey, R., Dore, P., Aichouch, M., and Ohayon, E. (2015). Blind hypervision to protect virtual machine privacy against hypervisor escape vulnerabilities. In *Industrial Informatics (INDIN), 2015 IEEE 13th International Conference on*.

Dévigne, C., Bréjon, J. B., Meunier, Q., and Wajsbürt, F. (2015). Executing secured virtual machines within a manycore architecture. In *2015 Nordic Circuits and Systems Conference (NORCAS) : NORCHIP International Symposium on System-on-Chip (SoC)*.

Références II

Dévigne, C., Bréjon, J.-B., Meunier, Q. L., and Wajsbürt, F. (2017). Executing secured virtual machines within a manycore architecture. *Microprocessors and Microsystems*. Extended papers from the 2015 Nordic Circuits and Systems Conference.

Inc., A. M. D. (2005). Amd64 virtualization codenamed "pacifica" technology : Secure virtual machine architecture reference manual.

Inc., A. M. D. (2008). Amd-v nested paging. Whitepaper : `http://developper.amd.com/assets/NPT-WP-11-final-TM.pdf`.

Jin, S., Ahn, J., Seol, J., Cha, S., Huh, J., and Maeng, S. (2015). H-svm : Hardware-assisted secure virtual machines under a vulnerable hypervisor. *IEEE Transactions on Computers*, 64(10) :2833–2846.

Keller, E., Szefer, J., Rexford, J., and Lee, R. B. (2010). Nohype : Virtualized cloud infrastructure without the virtualization. In *Proceedings of the 37th Annual International Symposium on Computer Architecture*, ISCA '10.

Marshall, D. (2007). Understanding full virtualization, paravirtualization, and hardware assist. *VMWare White Paper*.

Masti, R. J., Marforio, C., Kostiainen, K., Soriente, C., and Capkun, S. (2015). Logical partitions on many-core platforms. In *Proceedings of the 31st Annual Computer Security Applications Conference*, ACSAC 2015.

Neiger, G., Santoni, A., Leung, F., Rodgers, D., and Uhlig, R. (2006). Intel virtualization technology : Hardware support for efficient processor virtualization.

Nightingale, E. B., Hodson, O., McIlroy, R., Hawblitzel, C., and Hunt, G. (2009). Helios : Heterogeneous multiprocessing with satellite kernels. In *In Symposium on Operating Systems Principles*, SOSP'09.

Pratt, I., Fraser, K., Hand, S., Limpach, C., Warfield, A., Magenheimer, D., Nakajima, J., and Mallick, A. (2005). Xen 3.0 and the art of virtualization. In *Linux symposium*, volume 2, pages 65–78. Ottawa, Ontario, Canada.

PUB, F. (2004). Standards for security categorization of federal information and information systems.

Références III

Statista. Size of the cloud computing and hosting market market worldwide from 2011 to 2019 (in billion U.S. dollars). https://www.statista.com/statistics/500541/worldwide-hosting-and-cloud-computing-market/.

Szefer, J., Keller, E., Lee, R. B., and Rexford, J. (2011). Eliminating the hypervisor attack surface for a more secure cloud. In *In Computer and Communications Security*, CCS'11.

Szefer, J. and Lee, R. B. (2012). Architectural support for hypervisor-secure virtualization. In *Proceedings of the Seventeenth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS XVII.

Uhlig, R., Neiger, G., Rodgers, D., Santoni, A. L., Martins, F. C. M., Anderson, A. V., Bennett, S. M., Kagi, A., Leung, F. H., and Smith, L. (2005). Intel virtualization technology.

Woo, D. and Lee, H. (2008). Extending amdahl's law for energy-efficient computing in the many-core era. volume 41. IEEE Computer.