Architectures for Secure High-Performance Computing

Ayaz Akram
Department of Computer Science,
University of California, Davis
yazakram@ucdavis.edu

Poster: http://www.ayazakram.com/papers/yarch_poster.jpg

Abstract—**High performance computing (HPC) is moving away from traditional simulation and modeling to large scale computational problems involving large datasets. Sometimes this data can be sensitive, provided by third parties to HPC centers or individual researchers, and raises security concerns. This work aims to provide secure architectures focused on HPC centers keeping the performance loss to minimum.**

I. Scope of the Problem

The use of sensitive data sets in HPC centers raises security concerns and eventually leads to the problem of mistrust between data providers, compute providers and users of HPC resources [32]. This, sometimes, results into an implicit trade-off between security and service provision. Currently, even if secure enclaves are made available to protect the sensitive data sets, they are far from user friendly. In contrast, the use of hardware based TEEs (trusted execution environments) can provide a usable compute model which can also guarantee security from other users or un-trusted components in the HPC system. Different CPU manufacturers have presented different TEE solutions so far (e.g. Intel's SGX [3], AMD's SEV [24], and ARM's TrustZone [10]), however, none of them are targeted towards the use-case of HPC. In this work, we plan to propose TEE based (or TEE inspired) security solutions for high performance computing centers.

Before delving into the discussion on how are we trying to solve this problem, it is worth identifying some features which distinguish HPC from general purpose computing environments and their significance for secure architectures i.e. what restrictions these features impose on a secure architecture and if/how some of them can be leveraged to simplify the secure architecture design.

- HPC applications are mostly multi-threaded and have large working sets. This means that the secure environment should be capable of supporting multiple execution threads and should have minimal performance overhead even if the memory size that needs to be secured is large.
- HPC applications mostly scale across multiple nodes and rely on message passing run-times like MPI for communication across nodes. Thus, protection against physical attacks when multiple nodes are involved should be provided. The support for secure boot, remote attestation across multiple nodes at the same time should also be provided. None of the existing TEEs support this.
- HPC centres usually have high speed network interconnects (e.g. InfiniBand) between multiple nodes (can support 10s of GB/s bandwidth) and mostly rely on one-sided communication protocols like RDMA [28]. Protocols like RDMA (which bypass OS mostly), while provide performance benefits, raise new security threats because of their one-sided communication nature. At the same time the bypassing of OS provides an opportunity to exclude OS from the trusted computing base (or have less trust in the OS).
- HPC applications rely on limited types of I/O e.g. network I/O is mostly used to communicate with other nodes and even the disk accesses reduce to network I/O as (distributed) file systems are usually maintained on remote nodes. Moreover, the OS is mostly bypassed and I/O is handled in user-space libraries or run-times.
- Nodes in an HPC center are alloted to a single user at a particular time and only get multiplexed at granularity of large time intervals. This can enable easier mechanisms to provide user isolation guarantees.
- HPC systems have also started to integrate accelerators (like GPUs and FPGAs) to offload certain applications or parts of applications to those processing elements. This necessitates the inclusion of these processing elements into trusted computing base as well.
- HPC applications often rely on third party libraries and it might be harder for such applications to be modified or re-written to port them to a different secure execution programming model. Thus, the secure solutions should try to reduce the number of changes that might be needed in the workloads (or avoid recompilation).

Table I provides a taxonomy of different TEE features that some of the current TEEs provide and an HPC-centric TEE should provide. The missing pieces are the things we plan to work on in this project.

II. Solution

In order to explain how are we tackling the problem discussed above, we can take a look at Figure 1, which shows an example dual node HPC system (logically it can be extended

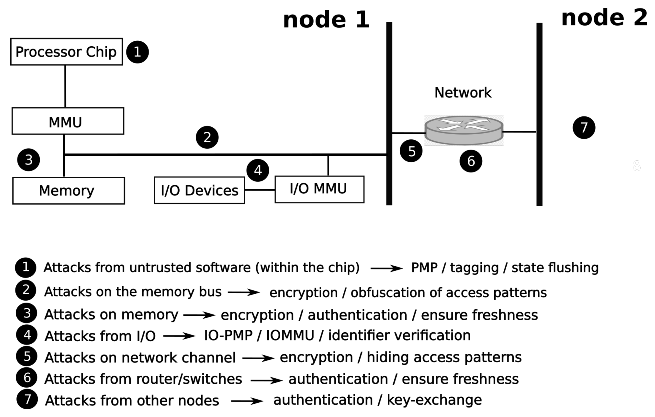
node 1
node 2
Processor Chip
MMU
Memory
I/O Devices
I/O MMU
Network
❶ Attacks from untrusted software (within the chip) → PMP / tagging / state flushing
❷ Attacks on the memory bus → encryption / obfuscation of access patterns
❸ Attacks on memory → encryption / authentication / ensure freshness
❹ Attacks from I/O → IO-PMP / IOMMU / identifier verification
❺ Attacks on network channel → encryption / hiding access patterns
❻ Attacks from router/switches → authentication / ensure freshness
❼ Attacks from other nodes → authentication / key-exchange

Fig. 1: Threat model of an HPC system (inspired from [36])

to more than 2 nodes), and identifies main attack points that can be exploited by the adversary. The possible protection mechanisms for each attack point are also shown as well (on the right side of pointers).

Of particular interest to HPC are: ❹, ❺, ❻, and ❼. These attack points have mostly not been considered by TEE developers in the past. The attacks at ❶, ❷, and ❸ are mostly already covered by the existing TEE solutions. However, as shown in Figure 1, there already exist a number of security mechanisms that can provide protection at these attack points as well. Essentially the solution to our problem boils down to coming up with the right combination of mechanisms that will provide protection for our threat model at lowest performance cost. Thus, we plan to perform a design space exploration of different available mechanisms and trimming it down to the most suitable mechanisms for HPC. At the same time, we plan to explore if we can come up with a single unified mechanism to provide protection against all of these attacks.

We already performed an extensive benchmarking of current TEE technologies to understand their performance implications and shortcomings [4], [6]–[8].

In this work, we rely on an academic proposal of TEEs, KeyStone [27], as it is open source and customizable. KeyStone [27] relies on RISC-V's primitives like PMP (physical memory protection) and allows platform specific extensions. KeyStone make use of machine mode (most privileged mode in RISC-V) based security monitor (SM), which can be entirely programmed in software, to control security mechanisms in the system. Furthermore, proposals of IO-PMP (physical memory protection for IO) are already in discussion. Using a RISC-V based TEE provides flexibility to easily extend or add new features in the ISA if needed. We plan to extend KeyStone to perform a design space exploration of the secure mechanisms referred above. Currently, KeyStone lacks a number of features that would make it ideal for such studies. For example, it only supports single threaded enclaves right now and is also expected to have performance implications while trying to synchronize PMP entries across multiple processors in a multiprocessor system. We plan to resolve these issues first.

III. Evaluation Methodology

We plan to rely on evaluations using qemu [13] and gem5 [14], [29]. KeyStone [27] has already been available on qemu for single node TEEs. We plan to use qemu for initial functionality testing and for more detailed studies we will be relying on gem5. As a first step, we will be porting KeyStone to gem5 [5]. We can also use CloudLab [20] for real system studies. As far as simulation of distributed systems is concerned, dist-gem5 [30] is a possible framework to use as well as researchers have already explored the use of gem5 integrated with network simulators like LogGOPSim [23] (e.g. as used by [22]).

Table I. Taxonomoy of different TEE features. **HPC centric** (row in green shade) refers to what is best for HPC. Brown shaded columns are of special importance from HPC perspective.

TEE	**Software Attacks**[1]				**Hardware Attacks**[2]		**Level**[3]	**TCB**	**I/O Handling**	**No Changes Needed**		**Use Cases**	**HPC Slowdown**[4]
	From processes	From OS/ hyper-visor	From I/O[8]	On I/O[8]	From I/O[8]	Physical Attacks				Hardware	Software		
SGX [18]	✓	✓	✓	✗	✗	✓	App.	App., CPU	outside enclave, in clear	✗	✗	Small desktop Apps.	large[5]
SEV [25]	✓	✓	✓	✗	✗	✓	VM	guest OS, App., CPU	using bounce buffers, in clear	✓	✓	VMs in Cloud	minimal[6]
TrustZone [10]	✓	✓	✓	✗	✗	✗	system parti-tion	App., trusted OS, CPU	I/O part of secure world/TCB	✗	✗	embedded	N/A
AWS Nitro [1]	✓	✗		✗	✗	✗	VM	VM, hy-pervisor	VM socket	✓	✗	VMs in cloud	minimal
KeyStone [27]	✓	✓	✓	✗	✗	✓	App.	App., RT, SM, CPU	outside enclave, in clear	✓	✗	variable	unclear[7]
HPC centric	✓	✓	✓	✓	✓	✓	App.	App.,CPU	secure	✓	✓	All	minimal

[1]Software attacks have software and [2]hardware attacks have hardware as the attack surface. [3]Level is the granularity/level at which protection is provided.
[4]No TEE supports multi-node trusted execution and use of software to create secure tunnel between TEEs on multiple nodes cause very high slowdown

[5]specially for multi-threaded and large memory Apps. [6]with careful memory allocation. [7]no support for multi-threaded enclave and has large slowdown for IO
Other Notes: These TEEs generally do not consider side channels. Threat of side channels depend on the data sensitivity and leakage rate.
Only SGX provides strong protection against integrity attacks. SEV-SNP provides some gaurantees against inegrity attacks. [8]I/O includes GPUs, accelerators and FPGAs as well

For our evaluations we plan to use HPC kernels like NAS Parallel Benchmark suite (NPB) [11], graph workloads like (GAPBS) [12] and other DOE HPC workloads (e.g. [2], [9], [17], [26].

IV. Related Work

There exist many other academic TEEs ([15], [16], [19], [33], [35]) apart from KeyStone. Similarly, there are various proposals to improve the performance implications of enclaves [21], [31], [34], [37]. However, none of them is focused on HPC and they do not target the aspects we are focusing on in this work.

References

[1] AWS Nitro Enclaves. https://aws.amazon.com/ec2/nitro/nitro-enclaves/.

[2] Hydrodynamics Challenge Problem, Lawrence Livermore National Laboratory. Technical Report LLNL-TR-490254.

[3] *Intel Software Gaurd Extensions (Intel SGX)*. Available: https://software.intel.com/en-us/sgx/details.

[4] Ayaz Akram. Setting up Trusted HPC System in the Cloud. https://arch.cs.ucdavis.edu/blog/2020-11-19-cloud-hpc, 2020.

[5] Ayaz Akram, Venkatesh Akella, Sean Peisert, and Jason Lowe-Power. Enabling design space exploration for risc-v secure compute environments. In *Fifth Workshop on Computer Architecture Research with RISC-V (CARRV 2021)*, pages 1–7, 2021.

[6] Ayaz Akram, Anna Giannakou, Venkatesh Akella, Jason Lowe-Power, and Sean Peisert. Using trusted execution environments on high performance computing platforms. In *Open-source Enclaves Workshop (OSEW 2019)*, july 2019.

[7] Ayaz Akram, Anna Giannakou, Venkatesh Akella, Jason Lowe-Power, and Sean Peisert. Performance analysis of scientific computing workloads on trusted execution environments. *arXiv preprint arXiv:2010.13216*, 2020.

[8] Ayaz Akram, Anna Giannakou, Venkatesh Akella, Jason Lowe-Power, and Sean Peisert. Performance analysis of scientific computing workloads on general purpose tees. In *Proceedings of the 35th IEEE International Parallel & Distributed Processing Symposium (IPDPS)*, May, 2021.

[9] Stephen F Altschul, Warren Gish, Webb Miller, Eugene W Myers, and David J Lipman. Basic Local Alignment Search Tool. *Journal of molecular biology*, 215(3):403–410, 1990.

[10] Tiago Alves and Don Felton. TrustZone: Integrated Hardware and Software Security. *Information Quarterly*, pages 18–24, 2004.

[11] David H Bailey, Eric Barszcz, John T Barton, David S Browning, Robert L Carter, Leonardo Dagum, Rod A Fatoohi, Paul O Frederickson, Thomas A Lasinski, Rob S Schreiber, et al. The NAS Parallel Benchmarks. *The International Journal of Supercomputing Applic ations*, 5(3):63–73, 1991.

[12] Scott Beamer, Krste Asanović, and David Patterson. The GAP Benchmark Suite. *arXiv preprint arXiv:1508.03619*, 2015.

[13] Fabrice Bellard. QEMU, a Fast and Portable Dynamic Translator. In *USENIX Annual Technical Conference, FREENIX Track*, pages 41–46. Anaheim, CA, Anaheim, CA, 10-15 April 2005.

[14] Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R Hower, Tushar Krishna, Somayeh Sardashti, et al. The gem5 Simulator. *ACM SIGARCH Computer Architecture News*, 39(2):1–7, May 2011.

[15] Ferdinand Brasser, David Gens, Patrick Jauernig, Ahmad-Reza Sadeghi, and Emmanuel Stapf. Sanctuary: Arming trustzone with user-space enclaves. In *NDSS*, 2019.

[16] David Champagne and Ruby B Lee. Scalable architectural support for trusted software. In *HPCA-16 2010 The Sixteenth International Symposium on High-Performance Computer Architecture*, pages 1–12. IEEE, 2010.

[17] Cy Chan et al. Mobiliti: Scalable Transportation Simulation Using High-Performance Parallel Computing. In *ITSC*, pages 634–641, 2018.

[18] Victor Costan and Srinivas Devadas. Intel SGX Explained. Cryptology ePrint Archive, 2016. https://eprint.iacr.org/2016/086.

[19] Victor Costan, Ilia Lebedev, and Srinivas Devadas. Sanctum: Minimal hardware extensions for strong software isolation. In *25th {USENIX} Security Symposium ({USENIX} Security 16)*, pages 857–874, 2016.

[20] Dmitry Duplyakin, Robert Ricci, Aleksander Maricq, Gary Wong, Jonathon Duerig, Eric Eide, Leigh Stoller, Mike Hibler, David Johnson, Kirk Webb, Aditya Akella, Kuangching Wang, Glenn Ricart, Larry Landweber, Chip Elliott, Michael Zink, Emmanuel Cecchet, Snigdhaswin Kar, and Prabodh Mishra. The design and operation of CloudLab. In *Proceedings of the USENIX Annual Technical Conference (ATC)*, pages 1–14, July 2019.

[21] Anders T Gjerdrum, Robert Pettersen, Håvard D Johansen, and Dag Johansen. Performance of Trusted Computing in Cloud Infrastructures with Intel SGX. In *Proceedings of the 7th International Conference on Cloud Computing and Services Science*, pages 668–675, Porto, Portugal, Apr. 2017.

[22] Torsten Hoefler, Salvatore Di Girolamo, Konstantin Taranov, Ryan E Grant, and Ron Brightwell. spin: High-performance streaming processing in the network. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–16, 2017.

[23] Torsten Hoefler, Timo Schneider, and Andrew Lumsdaine. Loggopsim: simulating large-scale applications in the loggops model. In *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, pages 597–604, 2010.

[24] David Kaplan. AMD x86 Memory Encryption Technologies. In *Linux Security Summit*, 2017.

[25] David Kaplan, Jeremy Powell, and Tomand Woller. AMD MEMORY ENCRYPTION, 2016. White paper.

[26] Adam J Kunen, Teresa S Bailey, and Peter N Brown. KRIPKE - A Massively Parallel Transport Mini-App. Technical report, Lawrence Livermore National Lab (LLNL), Livermore, CA (United States), 2015.

[27] Dayeol Lee, David Kohlbrenner, Shweta Shinde, Krste Asanović, and Dawn Song. Keystone: An open framework for architecting trusted execution environments. In *Proceedings of the Fifteenth European Conference on Computer Systems*, pages 1–16, 2020.

[28] Jiuxing Liu, Jiesheng Wu, and Dhabaleswar K Panda. High performance rdma-based mpi implementation over infiniband. *International Journal of Parallel Programming*, 32(3):167–198, 2004.

[29] Jason Lowe-Power, Abdul Mutaal Ahmad, Ayaz Akram, Mohammad Alian, Rico Amslinger, Matteo Andreozzi, Adrià Armejach, Nils Asmussen, Srikant Bharadwaj, Gabe Black, et al. The gem5 simulator: Version 20.0+. *arXiv preprint arXiv:2007.03152*, 2020.

[30] Alian Mohammad, Umur Darbaz, Gabor Dozsa, Stephan Diestelhorst, Daehoon Kim, and Nam Sung Kim. dist-gem5: Distributed simulation of computer clusters. In *2017 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 153–162. IEEE, 2017.

[31] Meni Orenbach, Pavel Lifshits, Marina Minkin, and Mark Silberstein. Eleos: ExitLess OS Services for SGX Enclaves. In *Proceedings of the 12th ACM European Conference on Computer Systems*, pages 238–253, Belgrade, Serbia, Apr. 2017.

[32] Sean Peisert. Security in high-performance computing environments. *Communications of the ACM*, 60(9):72–80, 2017.

[33] Ling Ren, Christopher W Fletcher, Albert Kwon, Marten Van Dijk, and Srinivas Devadas. Design and implementation of the ascend secure processor. *IEEE Transactions on Dependable and Secure Computing*, 16(2):204–216, 2017.

[34] Gururaj Saileshwar, Prashant J Nair, Prakash Ramrakhyani, Wendy Elsasser, Jose A Joao, and Moinuddin K Qureshi. Morphable Counters: Enabling Compact Integrity Trees for Low-Overhead Secure Memories. In *51st Annual IEEE/ACM International Symposium on Microarchitecture*, 2018.

[35] G Edward Suh, Dwaine Clarke, Blaise Gassend, Marten Van Dijk, and Srinivas Devadas. Aegis: Architecture for tamper-evident and tamper-resistant processing. In *ACM International Conference on Supercomputing 25th Anniversary Volume*, pages 357–368, 2003.

[36] Jakub Szefer. Principles of secure processor architecture design. *Synthesis Lectures on Computer Architecture*, 13(3):1–173, 2018.

[37] Meysam Taassori, Ali Shafiee, and Rajeev Balasubramonian. VAULT: Reducing Paging Overheads in SGX with Efficient Integrity Verification Structures. In *Proceedings of the 23rd ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 665–678, Williamsburg, VA, Mar. 2018.