Building Trustworthy Many-Core Systems

ABSTRACT

Future many-core computer architectures present two emerging trends; a move from general purpose cores to specialised cores, and a need for Operating System (OS) designs that embrace heterogeneity. Proposed many-core OS architectures have focused on performance and scalability, however few have considered trustworthiness in the traditional security sense (confidentiality, integrity, and availability).

In this work, we propose a design for many-core, heterogeneous systems that introduces architectural security primitives to improve the trustworthiness of future systems. Two main problems will be explored in this work. First, how can we leverage replication (e.g. of OS services) to increase the fault tolerance characteristics (i.e. availability) of many-core OSs in response to software and hardware failures? Second, how can we provide a unified trusted execution environment (for confidentiality and integrity of data) in a heterogeneous system? The end goal is a trustworthy system design for many-core systems that provides trusted services to each processing unit in the system, promotes high levels of fault tolerance (e.g. using state-machine replication), and allows rapid recovery in safety-critical systems such as autonomous vehicles and medical devices.

1 INTRODUCTION

The slowing of Moore's Law has changed the design requirements for Operating Systems (OSs) of the future. As we move away from general processing cores and towards more specialised processor technology [26], there is a demand for OS designs that support the native heterogeneity of the processors available to user applications [22]. While several designs for many-core operating systems have been proposed [2, 4, 5, 29], their primary focus has been improving performance and scalability, with little explicit attention to factors affecting trustworthiness. This is problematic for safety-critical systems of the future (e.g. smart-grids, autonomous vehicles, and medical devices) as factors such as reliability, fault tolerance, and security are considered key metrics for how safety-critical architectures are assessed [23].

Software bugs [8, 14, 16, 21] and hardware failures [19, 28] threaten the trustworthiness of current monolithic kernels. A significant majority of these faults can be attributed to device drivers [8], with the typical mitigation involving the introduction of a new kernel reliability component [7, 24, 25, 31] responsible for the detection and remediation of any kernel errors resulting from an extension. However, existing reliability mechanisms present a single point-of-failure, since there is typically one (software) reliability component responsible for many drivers. For safety-critical systems, Dunn [11] identifies such single points of failure as one of the primary risks to safety-critical systems.

In addition to reliability, recent research has explored extending the trust model of Trusted Execution Environments (TEEs) like Intel SGX enclaves to also include external devices like FPGAs [20], GPUs [27], and storage devices [13], with modern SmartNICs also shipping with ARM Trustzone support [6]. As we move towards heterogeneous systems that are not CPU-centric, TEEs cannot treat such models as 'extensions', but rather need to provide a unified TEE model of security primitives and abstractions to all software applications in the architecture. The key challenge we will be exploring is how to move from a single (CPU-based) hardware root-of-trust, towards a Trusted-Computing-Base (TCB) that is composed of many connected heterogeneous hardware devices that will form a distributed root-of-trust in the architecture.

We aim to address these gaps in heterogeneous, many-core systems research by exploring an OS architecture for safety-critical many-core, heterogeneous systems that focuses on trustworthiness. Unlike previous work, we use replication as a recoverability mechanism rather than solely for sharing state (e.g. capabilities like Barrelfish [4]), and assess if this enables faster recovery from hardware and software faults in an OS service or device driver in comparison to existing shadow-driver [24] based approaches like Redleaf [17, 18]. Our design will also consider a TEE model that treats devices like FPGAs as first-class citizens, in contrast to a CPU-centric scheme where they are treated as extensions. This will allow applications to directly interface with accelerators as part of their execution, rather than requiring kernel interference for all requests.

2 OVERVIEW OF THE PROPOSED WORK

We envision two broad contributions to the systems research community from the proposed work:

(1) A State Machine Replication (SMR) protocol for secure fail-over as a result of hardware and software failures on safety-critical, many-core architectures.
(2) Design of a unified TEE library that allows applications to communicate directly with TEEs running different implementation across a range of heterogeneous devices.

Each of these contributions form two symbiotic sub-projects. A unified TEE abstraction allows any process state to be securely stored during the SMR secure -fail-over protocol, and then securely transferred whenever the process has recovered. The system design considers the types of failures that are seen in safety-critical use-cases such as medical devices [1]. Initially, the scope for our fault model will focus on recovery aspects of hardware failures (e.g. cores losing power and being taken offline), and software issues (e.g. bugs in an I/O driver [9] that may prevent an alarm from sounding). We can then consider security aspects of the SMR scheme such as Byzantine Fault Tolerance later in the project.

SMR-based Recovery Mechanism. Our design provides a SMR-based reliability service for critical applications in the system. The goal of this service is to maintain a consistent view of an arbitrary OS service's state so that a replica can take over the instant a failure is detected in a monitored component. Unlike previous designs

EuroDW'21, April 2021, Edinburgh, Scotland, UK
2021.

such as shadow drivers [24], the recovery service does not restart and replay the driver operations, but instead directs application requests to an existing replica executing on a separate core. This design enables quick response times when a failure is detected, with the replication service providing a hot-reload style of recovery for a process.

The system's SMR scheme will as a starting point extend previous work on many-core consenus such as PaxosInside, a non-blocking, single-acceptor variant of Paxos [10, 12]. As with PaxosInside, in our initial design a proposer can be any OS component that shares its state with the acceptor. The acceptor (supported by a set of backup-acceptors) is the OS service for managing the current state value of critical processes and handling the secure-fail-over procedure. Finally, the replica core is modelled as a learner, recording all state information shared during a consensus round in an attempt to mirror the execution of the proposer OS service. The PaxosInside protocol does not however fully consider the interaction with kernel components used to manage recovery as we do in our work, nor does it consider potentially Byzantine behaviour which we aim to cover in later stages of the project.

Heterogeneous Trusted Execution. While research has explored how to extend the trust model of trusted execution environments to include an external component [13, 20] or moved execution to devices like a GPU [27], exisitng approaches have involved a CPU-centric kernel component (e.g. device driver) that places management responsibility on the CPU-centric host. Such an approach implies that all heterogeneous devices must trust the resulting kernel implementation to honestly provide services like memory-management for allowing TEEs to return data to user-space applications via the CPU. As architectures move towards an environment where the CPU is no longer the only hardware root-of-trust, TEEs will require abstractions and security primitives that explicitly support a model of distributed roots of trust.

Our approach considers that each processing unit (e.g., CPUs, GPU, and NICs) has the capacity to provide a TEE to user applications, such as a secure networking stack. The key challenges that we are considering from a heterogeneous, many-core system perspective is how will the abstractions provide attestation for each of the different TEE types, and how will the TEE be presented to the different system components (e.g., to transfer data between two TEEs). By creating a unified interface for each TEE type, there is no requirement on a TEE to place any trust in a kernel component like in a CPU-centric approach. This allows the TCB in a many-core environment to remove the OS from the threat model, as there is no longer a requirement to build drivers for communicating with external devices via the kernel.

3 WORK TO BE DONE

To address our first research problem surrounding availability, we will extend the replication mechanism currently available in Barrelfish. This will involve the design and implementation of a suitable SMR protocol and recovery management component that places a component replica on different cores, and the implementation of an OS service that uses our replication service. We will evaluate this approach experimentally by developing a buggy I/O (e.g., networking) driver, assessing metrics for protocol performance including latency and throughput of messages of various sizes/frequency, and time of acceptor and leader re-election. Research into our second research question surrounding trusted-execution for confidentiality and integrity will come later in the project. To design the unified library we will need to identify classes of TEEs (e.g., GPU, smartNICs) and the APIs that are required for cross-TEE communication (e.g., Intel SGX to Arm TrustZone). We will then develop an OS service that runs on a device like a GPU or smart NIC and explore how we can leverage our unified TEE library to allow applications to move secure computation to the device that best fits the workload.

By the end of this academic year I am aiming to have an early prototype of the replication service implemented on some many-core OS such as Barrelfish. The experiments required for the evaluation will then begin at the start of the following academic year. Following the findings from the first research question, we will begin work on addressing TEEs in many-core systems. We anticipate a further year and a half to work on the design, implementation, and evaluation of this design.

4 RELATED WORK

The earliest proposal for embracing CPU ISA heterogeneity was the multikernel design [4]. In this approach, they treat the OS as a distributed system where each CPU core is a node under a share-nothing approach. Each core then runs its own kernel and inter-core communication is made explicit using a message-passing protocol. Barrelfish uses 2-Phase-Commit (2PC) for sharing state between applications. Our work aims to extend from this, but instead of using replication to share state, we aim to use replication for the rapid recovery of system services.

Popcorn Linux [3] adopts the Linux kernel for execution in a heterogeneous (ISA) system with a replicated kernel design. However, as this still uses the monolithic Linux kernel, it promotes standard Linux abstractions that treat each accelerator as a device, rather than first class citizens. We aim to explore trustworthiness in an inclusive model that treats non-CPU devices as first-class citizens.

Recent work has highlighted the need for alternative kernel abstraction models for giving applications direct access to the underlying hardware components, for instance in kernel by-passing for data-centre networking [30], or using a composition design pattern for flexible kernel design [15], to allow (I/O) applications to access accelerators without going through the kernel. We focus on defining a set of architectural abstractions for heterogeneous many-core systems that enhances the security profile of each system component. Such abstractions will allow applications to define their own trust model for each accelerator in the system, rather than having to extend it from a CPU-centric trust model like Intel SGX.

Redleaf [17, 18] is a recent clean-slate design for providing component isolation and transparent recoverability using language-based mechanisms in Rust. Their reliability mechanism design is in part based on shadow drivers [24], using a proxy that monitors communications between the driver and user-space application, and replying the communications log to a new driver instance when failure is detected. Our reliability mechanism is based on a non-blocking SMR protocol, that aims to provide a hot-reload style recovery design.

REFERENCES

[1] Homa Alemzadeh, Ravishankar K. Iyer, Zbigniew Kalbarczyk, and Jai Raman. 2013. Analysis of Safety-Critical Computer Failures in Medical Devices. *IEEE Security & Privacy* 11, 4 (July 2013), 14–26. https://doi.org/10.1109/MSP.2013.49

[2] Nils Asmussen, Marcus Völp, Benedikt Nöthen, Hermann Härtig, and Gerhard Fettweis. 2016. M3: A Hardware/Operating-System Co-Design to Tame Heterogeneous Manycores. In *Proceedings of the Twenty-First International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '16)*. Association for Computing Machinery, New York, NY, USA, 189–203. https://doi.org/10.1145/2872362.2872371 event-place: Atlanta, Georgia, USA.

[3] Antonio Barbalace, Marina Sadini, Saif Ansary, Christopher Jelesnianski, Akshay Ravichandran, Cagil Kendir, Alastair Murray, and Binoy Ravindran. 2015. Popcorn: bridging the programmability gap in heterogeneous-ISA platforms. In *Proceedings of the Tenth European Conference on Computer Systems*. 1–16.

[4] Andrew Baumann, Paul Barham, Pierre-Evariste Dagand, Tim Harris, Rebecca Isaacs, Simon Peter, Timothy Roscoe, Adrian Schüpbach, and Akhilesh Singhania. 2009. The multikernel: a new OS architecture for scalable multicore systems. In *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles - SOSP '09*. ACM Press, Big Sky, Montana, USA, 29. https://doi.org/10.1145/1629575.1629579

[5] Silas Boyd-Wickizer, Haibo Chen, Rong Chen, Yandong Mao, M Frans Kaashoek, Robert Tappan Morris, Aleksey Pesterev, Lex Stein, Ming Wu, Yue-hua Dai, and others. 2008. Corey: An Operating System for Many Cores.. In *OSDI*, Vol. 8. 43–57.

[6] BROADCOM. 2018. *Stingray™ 2x25Gb High-Performance Data Center Smart NIC*. https://docs.broadcom.com/doc/PS225-PB

[7] Miguel Castro, Manuel Costa, Jean-Philippe Martin, Marcus Peinado, Periklis Akritidis, Austin Donnelly, Paul Barham, and Richard Black. 2009. Fast byte-granularity software fault isolation. In *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles - SOSP '09*. ACM Press, Big Sky, Montana, USA, 45. https://doi.org/10.1145/1629575.1629581

[8] Andy Chou, Junfeng Yang, Benjamin Chelf, Seth Hallem, and Dawson Engler. 2001. An empirical study of operating systems errors. In *Proceedings of the eighteenth ACM symposium on Operating systems principles*. 73–88.

[9] MITRE Corporation. 2018. *CVE-2018-1000004*. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-1000004

[10] Tudor David, Rachid Guerraoui, and Maysam Yabandeh. 2014. Consensus inside. In *Proceedings of the 15th International Middleware Conference on - Middleware '14*. ACM Press, Bordeaux, France, 145–156. https://doi.org/10.1145/2663165.2663321

[11] W.R. Dunn. 2003. Cover feature - Designing safety-critical computer systems. *Computer* 36, 11 (Nov. 2003), 40–46. https://doi.org/10.1109/MC.2003.1244533

[12] Rachid Guerraoui and Maysam Yabandeh. 2010. PaxosInside. (2010), 10. http://infoscience.epfl.ch/record/153309

[13] Robert Krahn, Bohdan Trach, Anjo Vahldiek-Oberwagner, Thomas Knauth, Pramod Bhatotia, and Christof Fetzer. 2018. Pesos: policy enhanced secure object store. In *Proceedings of the Thirteenth EuroSys Conference*. ACM, Porto Portugal, 1–17. https://doi.org/10.1145/3190508.3190518

[14] Zhenmin Li, Lin Tan, Xuanhui Wang, Shan Lu, Yuanyuan Zhou, and Chengxiang Zhai. 2006. Have things changed now?: an empirical study of bug characteristics in modern open source software. In *Proceedings of the 1st workshop on Architectural and system support for improving software dependability - ASID '06*. ACM Press, San Jose, California, 25–33. https://doi.org/10.1145/1181309.1181314

[15] James Litton, Deepak Garg, Peter Druschel, and Bobby Bhattacharjee. 2019. Composing Abstractions using the null-Kernel. In *Proceedings of the Workshop on Hot Topics in Operating Systems*. ACM, Bertinoro Italy, 1–6. https://doi.org/10.1145/3317550.3321450

[16] Shan Lu, Soyeon Park, Eunsoo Seo, and Yuanyuan Zhou. 2008. Learning from mistakes: a comprehensive study on real world concurrency bug characteristics. In *Proceedings of the 13th international conference on Architectural support for programming languages and operating systems*. 329–339.

[17] Vikram Narayanan, Marek S. Baranowski, Leonid Ryzhyk, Zvonimir Rakamarić, and Anton Burtsev. 2019. RedLeaf: Towards An Operating System for Safe and Verified Firmware. In *Proceedings of the Workshop on Hot Topics in Operating Systems*. ACM, Bertinoro Italy, 37–44. https://doi.org/10.1145/3317550.3321449

[18] Vikram Narayanan, Tianjiao Huang, David Detweiler, Dan Appel, Zhaofeng Li, Gerd Zellweger, and Anton Burtsev. 2020. RedLeaf: Isolation and Communication in a Safe Operating System. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*. USENIX Association, 21–39. https://www.usenix.org/conference/osdi20/presentation/narayanan-vikram

[19] Edmund B. Nightingale, John R. Douceur, and Vince Orgovan. 2011. Cycles, cells and platters: an empirical analysisof hardware failures on a million consumer PCs. In *Proceedings of the sixth conference on Computer systems - EuroSys '11*. ACM Press, Salzburg, Austria, 343. https://doi.org/10.1145/1966445.1966477

[20] Hyunyoung Oh, Adil Ahmad, Seonghyun Park, Byoungyoung Lee, and Yunheung Paek. 2020. TRUSTORE: Side-Channel Resistant Storage for SGX using Intel Hybrid CPU-FPGA. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. 1903–1918.

[21] Nicolas Palix, Gaël Thomas, Suman Saha, Christophe Calvès, Julia Lawall, and Gilles Muller. 2011. Faults in Linux: Ten years later. In *Proceedings of the sixteenth international conference on Architectural support for programming languages and operating systems*. 305–318.

[22] John Shalf. 2020. The future of computing beyond Moore's law. *Philosophical Transactions of the Royal Society A* 378, 2166 (2020), 20190061. Publisher: The Royal Society Publishing.

[23] R. Singh. 1999. A systematic approach to software safety. In *Proceedings Sixth Asia Pacific Software Engineering Conference (ASPEC'99) (Cat. No.PR00509)*. IEEE Comput. Soc, Takamatsu, Japan, 420–423. https://doi.org/10.1109/APSEC.1999.809632

[24] Michael M. Swift, Muthukaruppan Annamalai, Brian N. Bershad, and Henry M. Levy. 2006. Recovering device drivers. *ACM Transactions on Computer Systems* 24, 4 (Nov. 2006), 333–360. https://doi.org/10.1145/1189256.1189257

[25] Michael M Swift, Brian N Bershad, and Henry M Levy. 2003. Improving the reliability of commodity operating systems. In *Proceedings of the nineteenth ACM symposium on Operating systems principles*. 207–222.

[26] Neil Thompson and Svenja Spanuth. 2018. The decline of computers as a general purpose technology: why deep learning and the end of Moore's Law are fragmenting computing. *Available at SSRN 3287769* (2018).

[27] Stavros Volos, Kapil Vaswani, and Rodrigo Bruno. 2018. Graviton: Trusted Execution Environments on GPUs. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*. USENIX Association, Carlsbad, CA, 681–696. https://www.usenix.org/conference/osdi18/presentation/volos

[28] Guosai Wang, Lifei Zhang, and Wei Xu. 2017. What Can We Learn from Four Years of Data Center Hardware Failures?. In *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, Denver, CO, USA, 25–36. https://doi.org/10.1109/DSN.2017.26

[29] David Wentzlaff and Anant Agarwal. 2009. Factored operating systems (fos) the case for a scalable operating system for multicores. *ACM SIGOPS Operating Systems Review* 43, 2 (2009), 76–85. Publisher: ACM New York, NY, USA.

[30] Irene Zhang, Jing Liu, Amanda Austin, Michael Lowell Roberts, and Anirudh Badam. 2019. I'm Not Dead Yet!: The Role of the Operating System in a Kernel-Bypass Era. In *Proceedings of the Workshop on Hot Topics in Operating Systems*. ACM, Bertinoro Italy, 73–80. https://doi.org/10.1145/3317550.3321422

[31] Feng Zhou, Jeremy Condit, Zachary Anderson, Ilya Bagrak, Rob Ennals, Matthew Harren, George Necula, and Eric Brewer. 2006. SafeDrive: Safe and recoverable extensions using language-based techniques. In *Proceedings of the 7th symposium on Operating systems design and implementation*. 45–60.