128-bit Addresses for the Masses (of Memory and Devices)

Mathieu Bacou[1], Adam Chader[1], Chandana Deshpande[2], Christian Fabre[3], César Fuguet[3], Pierre Michaud[4], Arthur Perais[2], Frédéric Pétrot[2], Gaël Thomas[1], Eduardo Tomasi[2,3] *

[1]Télécom SudParis, IP Paris [2]Univ. Grenoble Alpes, CNRS, Grenoble INP†, TIMA [3]Univ. Grenoble Alpes, CEA, List [4]Inria, Univ. Rennes, CNRS, IRISA

***Abstract*—The ever growing storage and memory needs in computer infrastructures makes 128-bit addresses a possible long-term solution to access vast swaths of data uniformly. In this abstract, we give our thoughts regarding what this would entail from a hardware/software perspective.**

I. Introduction & Motivation

Given the pace at which data is produced and stored in data-centers worldwide, the amount of physical memory installed within the machines that have to answer users requests in the blink of an eye might well need more than 64 bits to be addressed in a foreseeable future [1]. In the same vein, the amount of memory available in supercomputers grows regularly, e.g. Frontier, the first exaflop computer, aggregates 9.2 petabytes of memory for a total storage capacity of 716 petabytes [2]. Should all these data be exposed through the load/store memory interface (e.g., byte addressable disks), this would amount to needing 60-bit physical addresses. This trend is also exemplified by Intel, who increased physical addressing bits from 46 to 52 (and virtual ones from 48 to 57) [3, 4]. Although we do not yet require 65 bits, if there is anything we learned from the move from 32- to 64-bit, it is that starting early is key [5]. Indeed, as advocated by Waterman et al. [6] when defining the 128-bit RISC-V instruction set architecture, relying on ad-hoc address extensions hacks again (e.g. Intel PAE [5]) is only going to be an operating system nightmare that will eventually disappear. We should rather consider directly moving to 128-bit.

Moving to 128-bit poses some interesting challenges. Going bottom-up, this first means that the general purpose registers of the processors as well as many internal buffers will grow to 128-bit. During the 32 to 64-bit transition, Moore's law quickly absorbed the increase in hardware complexity caused by the change. Unfortunately, the transition from 64 to 128 bit will happen in a very different context where Moore's law has dramatically slowed down and Dennard's scaling has long stopped [7]. Nevertheless, processor hardware is only the tip of the iceberg, and the whole hardware groundworks and software stacks will need to be looked at from a different angle given the change in perspective this brings.

Indeed, cloud and high-performance infrastructures have evolved to become distributed and heterogeneous systems: one machine now includes several CPU sockets interconnected to multiple local memory banks through complex Non-Uniform Memory Access (NUMA) networks. Each socket is itself a complex system with an integrated network-on-chip interconnecting the cores and many levels of caches. In addition, the storage system is a complex stack of layers with variable performance characteristics and features. To top it all off, each computer contains heterogeneous accelerators dedicated to specific workloads: GPUs, FPGAs, SmartNICs, TPUs, Variable Precision FPUs, etc.

Meanwhile, the system software remains a stack of competing layers, each based on tweaked generic heuristics and concepts that try to manage the same hardware resources and system objects. For example, data-centers use a virtualization stack made of a Linux kernel-based virtual machine (VM), on top of a QEMU/KVM (Linux kernel)-based hypervisor, all in an attempt to handle the distribution and the heterogeneity. To handle the hardware and software diversity, Linux has grown to 20 millions lines of code; and despite this large code base, the kernel does still not provide adequate abstractions for modern hardware. As a result, many libraries turn to bypassing the operating system (OS), for performance reasons and to provide more adequate abstractions to the users. Given this, our conclusion is that the situation is not sustainable in the long term.

In this abstract, we give our views on a way to address the challenges and opportunities that a 64- to 128-bit transition brings into large scale software-centric system infrastructures.

II. Hardware Perspective

Considering 128-bit addresses is bound to put additional pressure on actual hardware, at the very least because buses on which virtual address travel will become wider.

A. *Efficient 128-bit Microarchitecture*

Naively doubling the width of datapaths, registers and functional units will incur significant power, area and latency overheads [8]. Combined to the doubled footprint of pointers in memory, the transition to 128-bit is likely to undermine performance rather than improve it, at least from the processor perspective. Our intuition is that most 128-bit computation will be address calculation rather than "regular" integer arithmetic. Therefore, we envision a clustered microarchitecture [9] with a 64-bit cluster where 32- and 64-bit integer arithmetic will be performed, and a 128-bit cluster where address calculation and

*Alphabetical order. Funded by ANR project *Maplurinum* (ANR-21-CE25-0016). Corresp.: arthur.perais@univ-grenoble-alpes.fr

†Institute of Engineering Univ. Grenoble Alpes

128-bit arithmetic will be performed. This divide & conquer approach exposes full 128-bit support to software with limited hardware investment in the processor backend. A key difficulty will be to identify 32- and 64-bit instructions that participate in address calculation and steer them to the address cluster, as inter-cluster communication generally incurs latency [9].

Distinguishing addresses from integer in the microarchitecture also enables compression of internal processor structures that store addresses. Indeed, if the address cluster mostly manipulates addresses, we can expect higher value locality than if both addresses and "regular" integer were stored. By mapping high order bits of addresses to a limited number of region identifiers in a dedicated hardware mapping table [10], any structure manipulating addresses (e.g., address cluster Physical Register File, Branch Target Buffer tags and targets, VIVT cache tags, TLB tags) can be heavily compressed, at the cost of an indirection to regenerate the full 128-bit address.

B. *Memory Hierarchy*

Dealing with the NUMA effects is one of the most important challenges with 128-bit addressable space. Indeed, highly heterogeneous supercomputers and datacenters will implement multi-level cache-hierarchies with multiple distributed memory banks at the chip level to increase memory bandwidth. These memory banks are physically distributed but logically shared, that is, any device in the system can access any memory bank. However, the access latency depends on the physical distance between the device and the target memory bank, which is exacerbated with multi-socket and multi-board systems. A possible solution for reducing NUMA effects is to place code and data near the devices needing it. Unfortunately, correctly placing memory objects is hard when they are shared by multiple devices that are physically far from one another. While the OS can replicate code and data segments and put copies near the devices, this leads to yet another issue: the management of the copies. Data coherency and consistency must be guaranteed in case of multi-write and multi-read data. For example, CXL protocols [11] ensure cache coherency between a machine's devices, enabling extremely efficient direct accesses to memory-mapped registers. We envision OS-driven hardware mechanisms to ease both the replication and management of copies, for example by allowing the OS to dynamically define the cache coherency between devices.

III. Software & System Perspective

Our opinion is that the OS currently needs a redesign to handle the challenges of heterogeneity and disaggregation of the rack. This redesign can piggy-back on the –arguably– inevitable 64- to 128-bit transition.

Indeed, we advocate for redesigning the OS as a multi-kernel, centered around a large unified address space, and supported by RISC-V as a common minimal instruction set.

A. *Multi-kernel*

A multi-kernel is a distributed system: each processing unit runs a kernel, and they collaborate to execute processes that run on the different units. Multi-kernels have already been proposed [12, 13], but they only took into account plain homogeneous CPUs. They also limited themselves to one communication paradigm (message passing).

To expand the scope of previous works, we propose to revisit the concept of satellites: each heterogeneous device and accelerator is equipped with an optimized CPU solely dedicated to the operations of the control plane. By doing so, devices and accelerators can be seen as homogeneous processing units that are active in the disaggregated system. The common minimal instruction set and hardware virtualization will help in implementing satellite kernels that communicate efficiently via a unified 128-bits address space.

B. *Unified address space*

In the context of a disaggregated rack, the multi-kernel is a controller that grants access to different hardware resources to user tasks. We want to revisit the classic kernel interfaces and abstractions around the idea of direct access to the data plane, i.e., the memory. Indeed, all satellites will give out grants for other satellites to directly map some control structure and data buffers within their own memory.

This design is in line with the bypassing of the control plane to optimize data plane-related operations. This is also a way to homogenize the low-level interfaces of the kernel, all through a unified address space.

C. *Tooling*

From a more prosaic standpoint, and although most concepts can and will be demonstrated on 64 bits, we started to modify tools to support 128-bit software development [14]. Note that `gcc` has been supporting the `__int128_t`/`__uint128_t` types for decades, which made the actual work simpler than expected. Namely, we added preliminary rv128 support:

(1) in QEMU, most of it being upstreamed, with the notable exception of the elf128 file format we defined and the 16 KiB page tables we advocate for,
(2) in the GNU binutils, in particular the assembler `gas`, the linker `ld` and the debugger `gdb`,
(3) in the GNU compiler collection `gcc`, for the C language, rv128 target only.

The tools are far from being thoroughly tested yet, but are usable on simple software.

IV. Take Away

Current software centric infrastructure already supports a large amount of memory, many different devices for acceleration and storage, and suffers from the NUMA effect. The latter can only worsen with as compute farms grow, availability of byte addressable disks increases, and integration of application specific accelerators becomes widespread. Although it does not solve the issues at hand per se, a transition to 128-bit addressing can be leveraged to rethink the whole datacenter infrastructure stack.

References

[1] Matthew Wilcox. "Zettalinux: It's Not Too Late To Start". In: *Linux Plumbers Conference*. https://lpc.events/event/16/contributions/1223/. Dublin, Ireland, Sept. 2022.

[2] Tiffany Trader. *Top500: Exascale Is Officially Here with Debut of Frontier*. https://www.hpcwire.com/2022/05/30/top500-exascale-is-officially-here-with-debut-of-frontier/. May 2022.

[3] Intel Corporation. *5-Level Paging and 5-Level EPT*. https://www.intel.com/content/www/us/en/content-details/671442/5-level-paging-and-5-level-ept-white-paper.html?wapkw=5-Level%20Paging. May 2017.

[4] David L. Mulnix. *Third Generation Intel® Xeon® Processor Scalable Family On Two Socket Platform Technical Overview*. https://www.intel.com/content/www/us/en/developer/articles/technical/third-generation-xeon-scalable-family-overview.html. Feb. 2022.

[5] Robert R. Collins. "Paging Extensions for the Pentium Pro Processor". In: *Dr. Dobb's Journal Undocumented Corner* (July 1996). http://www.rcollins.org/ddj/Jul96/.

[6] Andrew Waterman et al. *The RISC-V instruction set manual, volume I: User-level ISA, Version 2.0*. EECS Department, UC Berkeley, Tech. Rep. https://github.com/riscv/riscv-isa-manual/releases/tag/isa-449cd0c. 2023.

[7] John L. Hennessy and David A. Patterson. "A new golden age for computer architecture". In: *Communications of the ACM* 62.2 (Jan. 2019), pp. 48–60.

[8] Victor Zyuban and Peter Kogge. "The Energy Complexity of Register Files". In: *Proc. of the international symposium on Low power electronics and design*. 1998, pp. 305–310.

[9] Richard E Kessler. "The alpha 21264 microprocessor". In: *IEEE micro* (1999), pp. 24–36.

[10] André Seznec. "Don't use the page number, but a pointer to it". In: *Proc. of the International Symposium on Computer Architecture*. Vol. 24. 2. 1996, pp. 104–113.

[11] CXL Consortium. *Compute Express Link: The Breakthrough CPU-to-Device Interconnect*. https://www.computeexpresslink.org/. 2020.

[12] Andrew Baumann et al. "The Multikernel: A New OS Architecture for Scalable Multicore Systems". In: *Proc. of Symposium on Operating Systems Principles*. Oct. 11–14, 2009, pp. 29–44.

[13] Edmund B. Nightingale et al. "Helios: Heterogeneous Multiprocessing with Satellite Kernels". In: *Proc. of Symposium on Operating Systems Principles)*. Oct. 11–14, 2009, pp. 221–234.

[14] Frédéric Pétrot, Sylvain Nory, and Juan Jose Garcia Duarte. *riscv-gnu-toolchain supporting the 128-bit extension*. https://github.com/fpetrot/riscv-gnu-toolchain. Aug. 2022.