Extremely Heterogeneous Systems – Not Just For Niches

Hermann Härtig, Nils Asmussen, Jeronimo Castrillon, Adam Lackorzynski, Michael Roitzsch, Carsten Weinhold, Akash Kumar

TU Dresden Department of Computer Science

Like many others, the authors believe that there will be increasing heterogeneity in execution units in the widest sense (e.g., CPUs, GPUs, DSPs, and FPGAs), memory technologies (e.g., cache hierarchies, scratchpad, and non-volatile RAM), and interconnect technologies (e.g., NoC, optical waveguides on interposers, memory stacking, and switches). Some of these may be based on recent technological developments (e.g., optical interconnects and FEFET), others on future break throughs (e.g., CNTFETs, wireless, DNA-based self assembly, plasmonic, spin-orbit, and THz). We believe, *the* major challenge for the future will be to enable such advanced technologies to serve more applications than just smaller and smaller niches.

We are aware that computers with fundamentally different operation modes are under research and development, like quantum computing, memristor-based neural computing, in-memory computing and the likes. We are skeptical that these developments will be mature enough in the time frame addressed by the workshop (operational deployment 2025 - 2040) in ways other than as serving a niche, potentially attached as (yet another) accelerator to some classically deterministic computer. We believe we must undergo a significant effort on making increasingly heterogeneous systems available for diverse applications including – to some extent – legacy ones, leading to the following challenges for operating systems (OSes), runtimes, tools, etc.:

Orthogonality We need to treat the areas of heterogeneity (execution units, memory, interconnects) as orthogonal, such that changes in one area do not require fundamental changes or complete conceptual redesigns in the other areas. Here are two examples of current approaches: a) Add "from the outside" something like virtual memory in a way that memory organization is made transparent to execution units (similar to IOMMUs and OpenCAPI). b) Provide means to separate core algorithms from their memory access patterns [2].

Interaction We need to enable – at a conceptional level – the technology-independent interconnection of heterogeneous execution units in ways that allow cross-unit interaction with only minimal or no help by complex general purpose CPUs. One consequential challenge is to avoid that all execution units need complicated architectural properties of current general purpose CPUs like virtual-memory support and user-kernel mode. Ideally, such architectural properties should be "transparently attached from the outside". A viable approach seems to be to add rather simple hardware components to the execution and memory units that support safe non-OS-dependent interaction and build an OS around it (M^3 [1]).

Isolation With extremely heterogeneous execution units – as a matter of fact, already with FPGAs – it will be next to impossible to assure tolerable or even correct (in the widest sense) behavior. The probabilistic properties of some recently proposed devices add evidence to that observation. Therefore we expect, effective isolation that still allows versatile communication will be a key point in future architectures. We believe that OS-level capability mechanisms can provide the conceptual model, but should be supported by interconnects to avoid expensive OS intervention.

Abstraction We assume that sheer numbers and the expected range of heterogeneity will make it hard for application designers to fully exploit them just by "hand design". We believe a solution to that problem can be built around a combination of these components: a) abstractions that hide the complexities of such heterogeneous systems from the application developer, for example domain-specific languages allow compilers to automatically map tasks to execution units, b) versatile monitoring plug-ins with near zero distortion, c) distributed bulletin board with (incomplete) information on the current global state (for example based on gossip), and d) fairly decentralized decision procedures for dynamic allocation of activities to memory and networks in the normal operational case. Occasional global system reorganizations should be constrained to very rarely happening situations.

We believe, a core observation (valid already today) is that no single one-fits-all runtime or tool chain will be sufficient, however all runtimes should honor the same carefully designed interface between application-specific runtimes and an OS that is aware of global state. We do not share the widely held believe in the HPC community that containers and variants of virtual-machine technology – both abstractions based on fairly conventional HW/ISA – are the best choice for an OS/runtime interface in general and especially not as foundation for extremely heterogeneous systems. Rather, we believe that a much simpler abstraction modeled after micro kernels like L4, M^3, and Barrelfish are much more promising.

We believe that data-flow task models and the bulk synchronous execution model must be among the supported models. We also strongly believe, that the interaction between the OS/runtime/toolchain community and the technology/architecture community should be improved. For example, a technology-spotting activity should be installed that forces the technology community to occasionally relay their thoughts. OS researchers need at least a rough idea of the properties expected from these new material developments regarding capacity, energy consumption, speeds and the like. Otherwise, the time from invention of new technologies to their application-level exploitation will be even longer than it is today.

Acknowledgments

This work is funded by the German priority program 1648 "Software for Exascale Computing" via the research project FFMK and the German Research Council DFG through the Cluster of Excellence Center for Advancing Electronics Dresden (cfaed [3]).

References

[1] Nils Asmussen, Marcus Völp, Benedikt Nöthen, Hermann Härtig, and Gerhard Fettweis. 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*, pages 189–203. ACM, 2016.

[2] Tal Ben-Nun, Ely Levy, Amnon Barak, and Eri Rubin. Memory access patterns: The missing piece of the multi-gpu puzzle. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '15, pages 19:1–19:12, New York, NY, USA, 2015. ACM.

[3] Jeronimo Castrillon et. al. A hardware/software stack for heterogeneous systems. *IEEE Transactions on Multi-Scale Computing Systems*, November 2017.