# Efficient and Scalable Core Multiplexing with M³v

Nils Asmussen, Sebastian Haas, Carsten Weinhold, Till Miemietz, Michael Roitzsch
Barkhausen Institut, Dresden, Germany

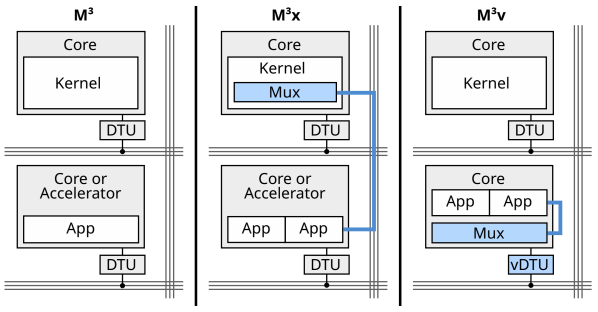

Figure 1: Tile multiplexing on M³ (no multiplexing), M³x (all tiles can be multiplexed using a single centralized OS tile), and M³v (general-purpose tiles can multiplex themselves).

M³ [1] is a hardware/software co-design that addresses the trend towards increasingly heterogeneous systems. It is based on a tiled hardware architecture and allows users to easily utilize general-purpose cores and special-purpose accelerators in a unified manner. Communication between tiles is achieved with a custom per-tile hardware component called data transfer unit (DTU). The DTU provides a uniform interface to all tiles, which simplifies heterogeneous systems. Additionally, the DTU isolates tiles from each other, because cross-tile communication is denied by default. Communication channels between tiles are set up by the M³ kernel, which runs on a dedicated OS tile. After the setup, applications can communicate directly via their DTU, bypassing the kernel. Therefore, we call such communication *fast-path communication*.

As depicted in Figure 1 (left), M³ does not support tile multiplexing and is therefore limited to one application per tile. The inability to multiplex tiles inhibits tile utilization when applications are occasionally idle. Multiplexing tiles among multiple applications therefore enables increased tile utilization. One reason that M³ is limited to one application per tile is that tile multiplexing impedes fast-path communication: the OS is responsible for tile multiplexing, but the goal of fast-path communication is to bypass the OS. For example, if an application is waiting for an incoming message, the OS needs to suspend the application to allow forward progress for other applications on the same tile. Later, the OS needs to resume the suspended application upon message arrival.

M³x [2] resolved this challenge in a manner that allows for multiplexing of both general-purpose cores and special-purpose accelerators. With M³x, the kernel performs all context switches on all tiles in the system remotely, as illustrated in Figure 1 (middle). Namely, the kernel is responsible for scheduling decisions, asks other tiles to save or restore their state, and switches between contexts. M³x retains fast-path communication if the recipient is running, but otherwise resorts to *slow-path communication*, which redirects the communication over the kernel. When two applications share a tile and cause frequent slow-path communication, M³x suffered from performance problems.

In this talk, we introduce M³ and then focus on M³v [3], a new core-multiplexing approach for the M³ system that replaces the general mechanism of M³x by a specific one for general-purpose cores. We trade some of the isolation and generality of M³x for improved efficiency. As sketched in Figure 1 (right), our design is based on 1) a core-local software multiplexer, which performs context switches on this core without involving the kernel on the OS tile and on 2) hardware virtualization of the DTU (*vDTU*). In contrast to the previous M³ prototypes that were simulated, we built a hardware FPGA-based implementation of M³ including our core-multiplexing support. In the evaluation, we compare to M³x in simulation using a context-switch heavy workload, showing a two-fold performance improvement and almost linear scalability up to 12 tiles for M³v, whereas M³x does not scale to two tiles. Additionally, we evaluate the performance of M³v in comparison to Linux on an FPGA platform, showing that M³v is competitive with single and multiple applications per core.

## References

[1] Nils Asmussen, Marcus Völp, Benedikt Nöthen, Hermann Härtig, and Gerhard Fettweis. M3: A hardware/operating-system co-design to tame heterogeneous manycores. International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), pages 189–203. ACM, 2016.

[2] Nils Asmussen, Michael Roitzsch, and Hermann Härtig. M³x: autonomous accelerators via context-enabled fast-path communication. USENIX Annual Technical Conference (ATC), pages 617–632. USENIX, 2019

[3] Nils Asmussen, Sebastian Haas, Carsten Weinhold, Till Miemietz, and Michael Roitzsch. Efficient and Scalable Core Multiplexing with M³v. International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), pages 452–466. ACM, 2022