Operating Heterogeneous Systems at Scale

Matthias Hille Nils Asmussen Pramod Bhatotia Hermann Härtig
Technische Universität Dresden
{matthias.hille,nils.asmussen,pramod.bhatotia,hermann.haertig}@tu-dresden.de

Background

The continuity of Moore's law provides chip designers with an ever growing amount of transistors. These can be used to put more cores on a single chip in order to gain performance through parallelism. However, this is not the most energy efficient approach to speed up computation. Yet, the power budget is a limiting factor for a chip's obtainable performance. Specialized accelerators provide better energy efficiency and execute the tasks they are designed for faster than traditional multicores. Already existing approaches like integrated GPUs, cryptographic co-processors or heterogeneous multicores like ARM big.LITTLE confirm this observation. In these systems the processing elements (PE) are typically connected via a network-on-chip (NoC), which shifts their internal communication characteristics towards those of distributed systems. Even today's multicore systems possess such properties when looking at their memory accessibility. The memory modules are split across the cores, hence accessing a memory location connected to a remote core is a remote operation using a different core's memory controller. However, the true nature of the chip's structure is not presented to the OS. Revealing the NoC to the OS enables a new approach to provide isolation.

The M^3 system [1], a recent work of our department, introduces isolation at the NoC level. This approach requires that PEs can only communicate and access memory via the NoC. The access to the NoC is provided by a data transfer unit (DTU), a small piece of hardware attached to each PE, which eliminates the necessity of MMUs or privileged mode to ensure isolation. The configuration of the DTU is performed remotely by a kernel, running on a dedicated PE. Thereby PEs without OS support, i.e. accelerators, can be easily integrated in a NoC-isolated architecture. At the moment the system is managed by a single microkernel instance, limiting the number of maintainable PEs.

Approaching Scalability

Divide and Conquer. Our poster is about a solution to scale the M^3 operating system to a large number of heterogeneous PEs. Therefore, we investigate on how to distribute kernel data structures in the M^3 system. By employing techniques known from distributed systems, e.g. coordination mechanisms and routing algorithms, we are assembling a set of microkernels to manage the system. Each kernel is in charge of a group of PEs, that is all kernel data related to PEs of one group is maintained by the corresponding kernel. However, PEs can interact with every other PE in the system. For that, a lookup technique, inspired by peer-to-peer overlays, allows a kernel to efficiently locate and access data managed by another kernel.

Consistent Permissions. An important aspect of our approach is the management of capabilities, which are used to represent applications' permissions. We are developing mechanisms to manage distributed capabilities in a flat capability space. When manipulating capabilities, e.g. revoking or inheriting, care has to be taken in order to maintain the capability space's consistency, since capabilities may be spread across kernels. This requires the close interaction of data distribution and lookup approaches to enable efficient capability management.

Kernel Load Balancing. Some applications demand OS services more frequently than others, hence our system's group composition of PEs is not static. Furthermore, the communication patterns of applications matter, since OS services involving PEs of other groups incur additional overhead. In order to minimize such group-spanning operations a kernel can transfer the responsibility for one or more PEs to another instance or even create a new kernel instance if all kernels are heavily loaded.

References

[1] N. Asmussen, M. Völp, B. Nöthen, H. Härtig, and G. 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*, ASPLOS '16, pages 189–203, New York, NY, USA, 2016. ACM.