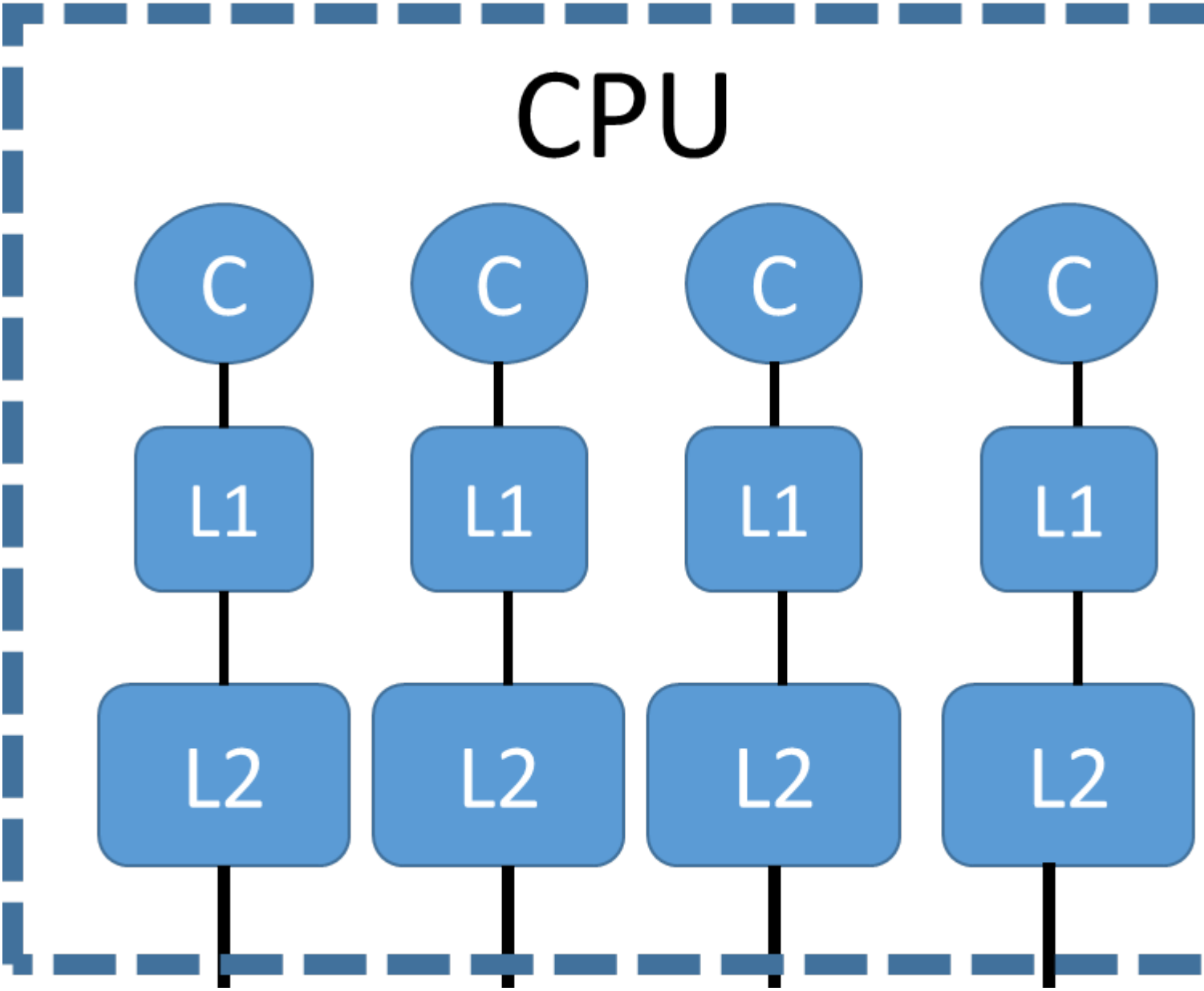
Memory hierarchy for GPU acceleration



Heterogeneous architectures have become the norm in the field of High Performance Computing. Many of such systems combine Graphics Processing Units (GPUs) and traditional multi-core processors to achieve many Teraflops of computational power.

Summary

GPU programming has traditionally been a cumbersome effort due to the need for explicit data movement between host and device. Vendors like NVIDIA and AMD have acknowledge this problem, and have incrementally provided mechanisms to simplify the task. The first step was to provide a unified view of the virtual memory space to allow pointers to be seamlessly shared between host and device. The latest release of the CUDA programming model goes even further by allowing concurrent access to shared data structures and dynamically performing page migration from the two pools of memory (DDR on host, GDDR on device). Unfortunately, the cost of resolving page faults at the GPU and starting an on-demand page migration is very high, and therefore any programmer attempting to maximize performance is still required to manually manage data movement.

Objectives

We are interested in improving the current situation by exploring mechanisms to perform on-demand low-overhead data migration to and from the GPU.
We are working with the gem5-gpu simulator, a combination of gem5 and gpgpu-sim, the two more widely used architecture simulators in the research community. A first step would be to implement in the simulator the current state-of-art of dynamic page migration found on CUDA 8 and Pascal-based GPUs and to validate the results against real hardware. This changes could serve as a baseline for future exploration of novel techniques, and would provide the student with a strong background to continue research on the topic either for a Master Thesis or further on for a PhD program.
Barcelona Supercomputing Center - Centro Nacional de Supercomputación

---