Efficient Joins on Heterogeneous Processors

Henning Funke, Sebastian Breß, Stefan Noll, Jens Teubner
DBIS Group, Dept. of Computer Science
TU Dortmund University
Germany
firstname.lastname@cs.tu-dortmund.de

ABSTRACT

Coprocessor hardware like graphics processors is frequently used to accelerate computations. Many applications achieve an increased throughput when processing data that is cached on the coprocessor, compared to the host processor alone. This does usually not apply to real world applications for two reasons. Firstly data sizes often exceed a coprocessor's memory capacity. Secondly it is desirable to compute on all available processors simultaneously and to manage the load condition, the work should be distributed evenly between processors. For both reasons extensive data transfers are necessary, which has lead to blaming the bus systems to be the dominating bottleneck for coprocessing. Especially for memory intensive workloads like in database query processing the advantages can marginalize when data transfers and memory accesses are not used carefully.

We investigate this set of problems in the context of databases by executing joins on multiple heterogeneous processors. We run a single pass join concurrently on CPU and GPU that is based on state-of-the-art hashing algorithms. We achieve an optimal load balancing and find that additional heterogeneous processors can substantially increase the overall throughput. Even when joining data sizes beyond a coprocessors memory capacity, throughput is rarely limited by the PCIe link.

When executing the join jointly on GPU and CPU, we observed an effect that has not been discussed in previous work. As shown in Fig. 2, the overall performance *degrades* as we add more CPU resources to compute the join. We attribute this to memory transfers between host and coprocessor, that encounter a race condition on main-memory. When executing a join concurrently on GPU and CPU, the overall throughput degrades, when the host system is put under stress (see Figure 1 and 2). The CPU is prioritized and steals bandwidth from the GPU although more GPU utilization would be more efficient. We consider main-memory bandwidth as performance factor for coprocessing and aim to find ways to optimize resource utilization.

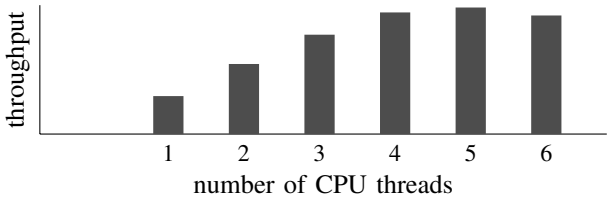

Fig. 1: Probe throughput of a multithreaded hash join on a CPU. The performance scales well to the number of 4 physical cores.

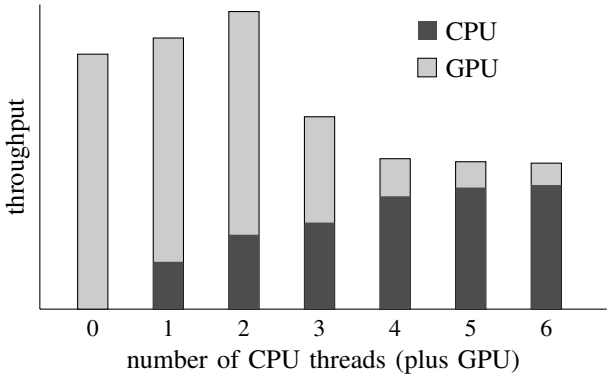

Fig. 2: Probe throughput of a join running concurrently on GPU and CPU. For a small number of CPU threads, the performance scales well, but when increasing the load we experience a degradation of coprocessor performance.

REFERENCES

[1] T. Kaldewey, G. Lohman, R. Mueller, and P. Volk, "GPU Join Processing Revisited," in *Proceedings of the Eighth International Workshop on Data Management on New Hardware*. ACM, 2012, pp. 55–62.

[2] C. Gregg and K. Hazelwood, "Where is the data? Why you cannot debate CPU vs. GPU performance without the answer," in *Performance Analysis of Systems and Software (ISPASS), 2011 IEEE International Symposium on*. IEEE, 2011, pp. 134–144.

[3] P. W. Frey, R. Goncalves, M. Kersten, and J. Teubner, "Spinning Relations: High-Speed Networks for Distributed Join Processing," in *Proceedings of the Fifth International Workshop on Data Management on New Hardware*. ACM, 2009, pp. 27–33.

[4] D. A. F. Alcantara, *Efficient Hash Tables on the GPU*. University of California at Davis, 2011.