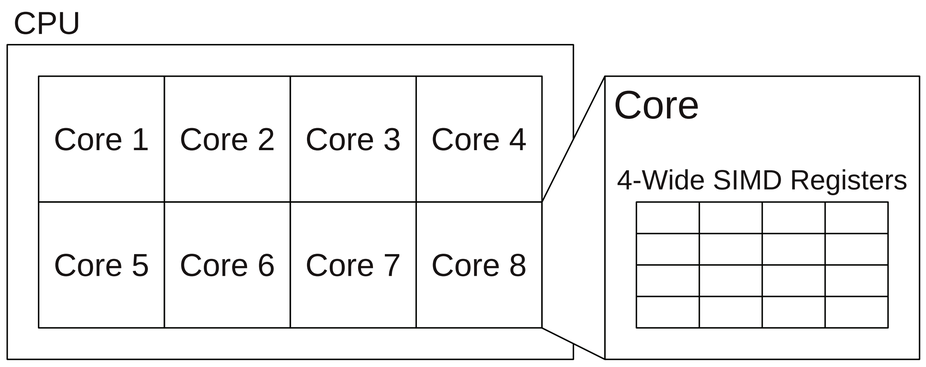
Hybrid Computing for HPC Applications

Sujin Philip, Brian Summa, Valerio Pascucci

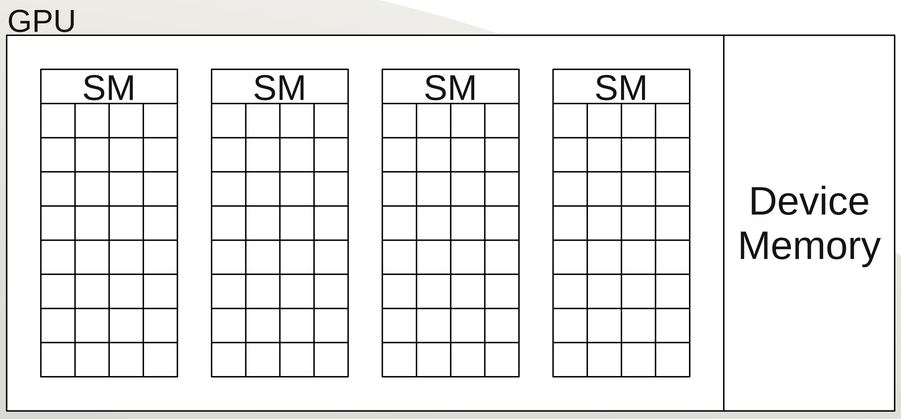
Motivation

- Clusters of commodity hardware are the most ubiquitous form of computing power available today.
- The processing power of such clusters has been consistently increasing.
- This performance increase is not only due to the increase in the number of connected nodes but also due to the increase in available parallelism within the nodes, in the form of multicore CPUs and GPUs.
- Hybrid computing is the efficient use of all these different types of resources to achieve high performance.

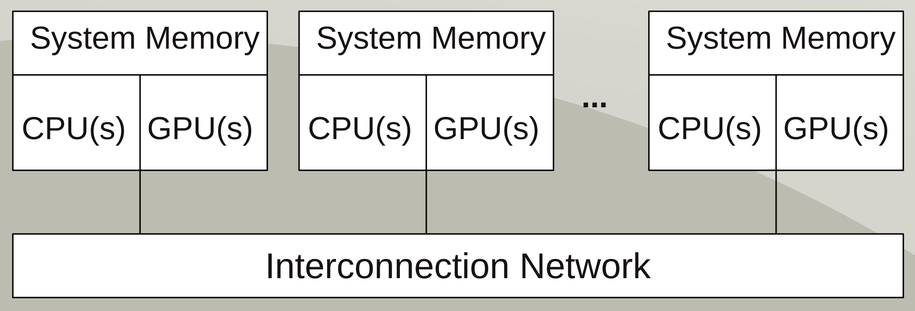
Hardware Architectures



A multicore CPU consists of many large, powerful cores. Within each core there is further parallelism available in the form of SIMD instructions and registers.



A cuda enabled GPU consists of multiple independent Streaming Multiprocessors (SM) each containing many smaller processors that execute in a SIMD fashion.



Cluster

A typical cluster consists of multiple nodes of computers working in parallel and connected to each other via a fast interconnection network.

Shared and Distributed Memory Systems

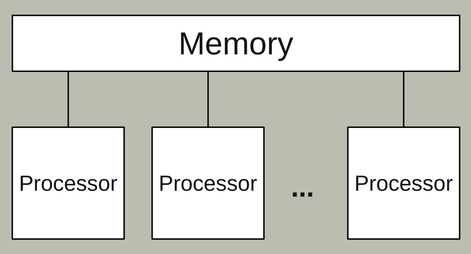


Fig: Shared Memory System

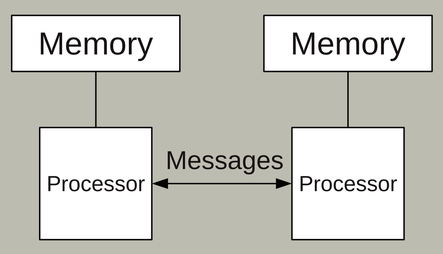


Fig: Distributed Memory System

Based on the hardware architecture, the programming paradigms for parallelism on these systems also differ.

Our Approach

Any computation can be broken down into a set of tasks. A dependency graph is created of these tasks with the edges representing the dependencies between them. Compilers may be able to generate these graphs automatically.

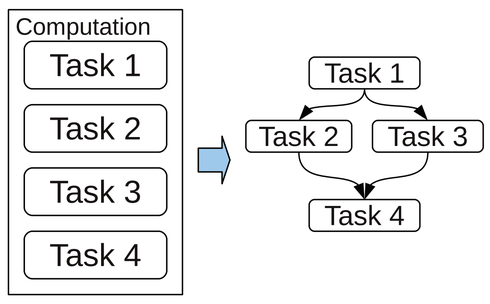


Fig: Computation and Task Graph

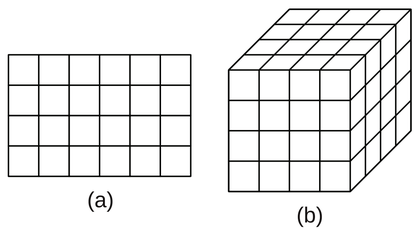


Fig: Data divided into chunks

Similarly, most large data can be divided into smaller chunks and processed independently.

A scheduler [3] analyzes the graph and schedules the tasks and data chunks for execution on the various available resources. A good scheduler schedules the data and task among the processors so as to achieve a balanced load.

Types of Parallelism

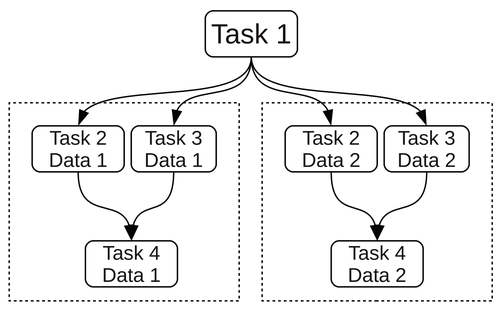


Fig: Data Parallelism

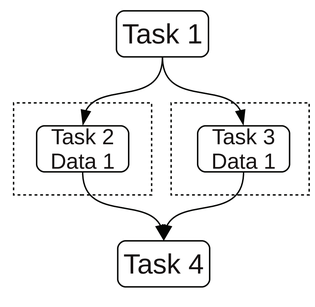


Fig: Task Parallelism

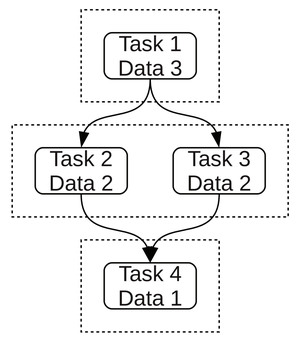
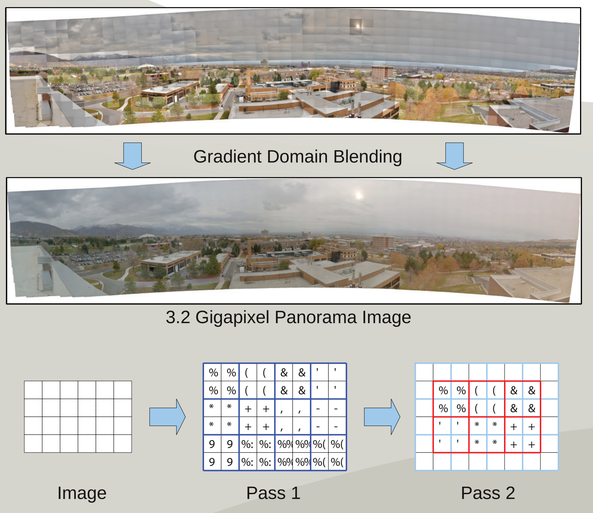


Fig: Pipeline Parallelism

With this design we are able to take advantage of three fundamental types of parallelism: data, task and pipeline parallelism.

Example



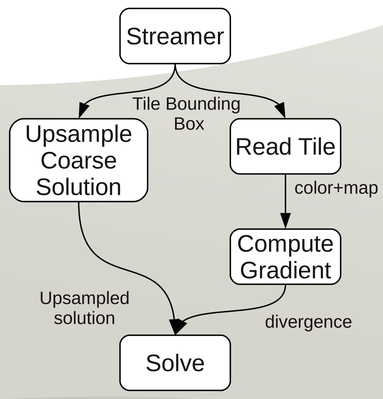


Fig: Task Graph

We have implemented a Hybrid solver for Gradient Domain Blending of massive panoramic images [1][2].

References

[1] Philip S., Summa B., Bremer P. T., Pascucci V.: Parallel Gradient Domain Processing of Massive Images. In Proceedings of the Eurographics Symposium on Parallel Graphics and Visualization 2011, pp. 11-19.

[2] Sujin Philip, Brian Summa, Peer-Timo Bremer and Valerio Pascucci: Hybrid CPU-GPU Solver for Gradient Domain Processing of Massive Images. To appear in International Conference on Parallel and Distributed Systems 2011.

[3] Vo H. T., Osmari D. K., Summa B., Comba J. L. D., Pascucci V., Silva C. T.: Streaming-enabled parallel dataflow architecture for multicore systems. Comput. Graph. Forum 29, 3 (2010), 1073-1082.

SCI INSTITUTE • EXHIBIT • EXPLORE • EXCITE • EXPERIENCE • EXCHANGE • SCIX

THE UNIVERSITY OF UTAH

SCI www.sci.utah.edu